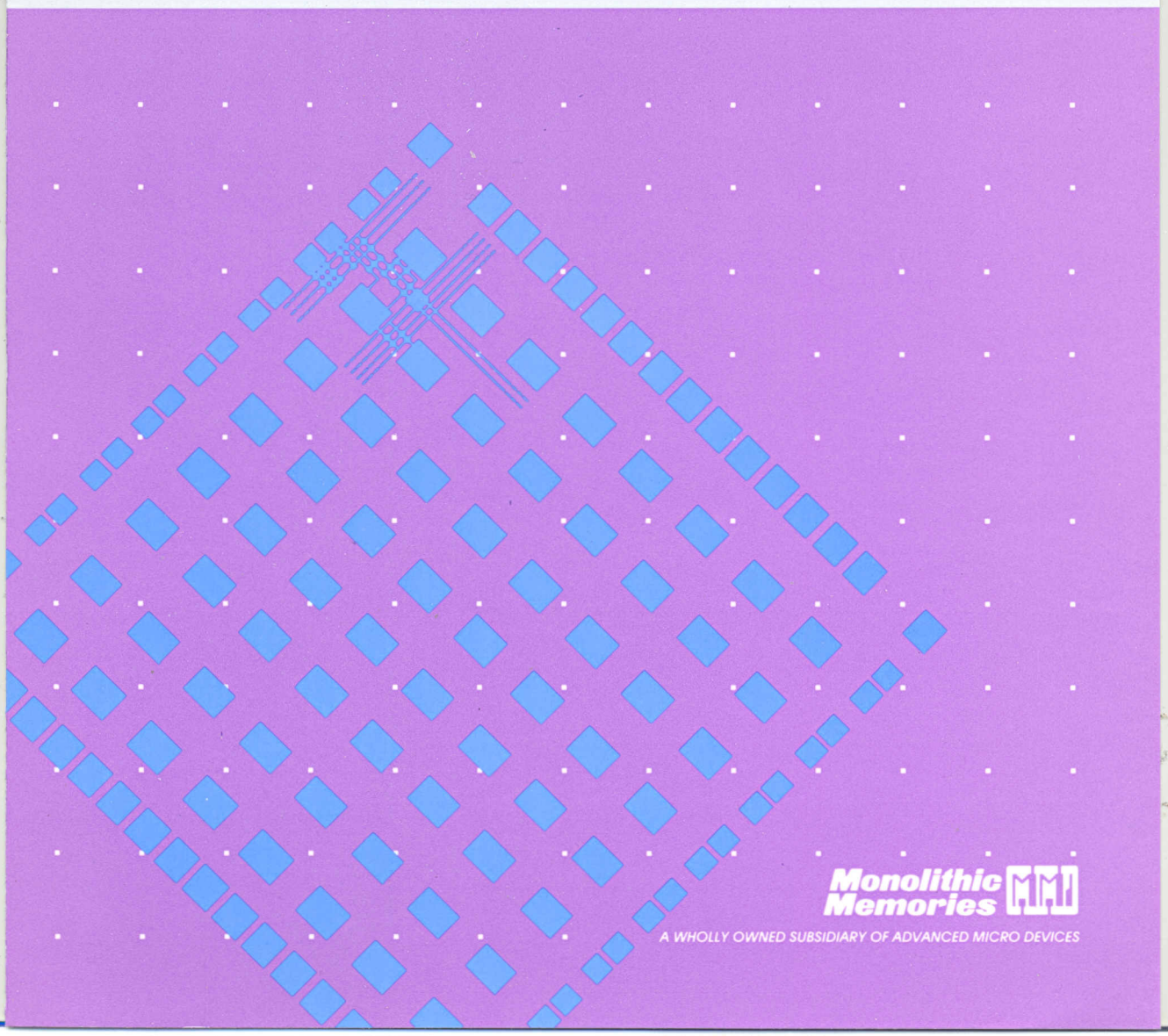
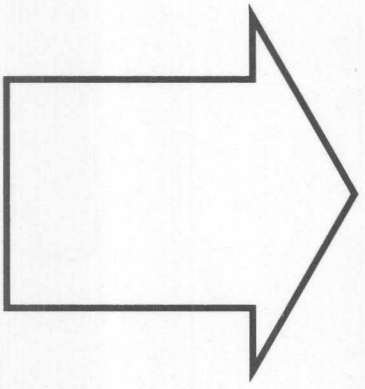
LCA
Applications Handbook

FIRST EDITION



Monolithic Memories

A WHOLLY OWNED SUBSIDIARY OF ADVANCED MICRO DEVICES



LCA APPLICATIONS HANDBOOK

FIRST EDITION

Monolithic Memories AMD

A Wholly Owned Subsidiary of Advanced Micro Devices

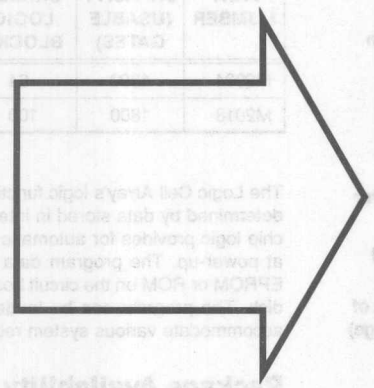
LCA Applications Handbook

Table of Contents

1

Index by Number

Index by Author



Logic Cell™ Array M2064/M2018

Features/Benefits

- CMOS programmable Logic Cell Array (LCA) for replacement of standard logic
- Completely reconfigurable by the user in the final system
- High performance equivalent to TTL SSI/MSI
  - 33 MHz flip-flop toggle rate (-33 speed grade)
  - 50 MHz flip-flop toggle rate (-50 speed grade)
  - 70 MHz flip-flop toggle rate (-70 speed grade)
- User configurable logic functions, interconnect and I/O for maximum flexibility
- 64/100 user-Configurable Logic Blocks (CLBs) providing usable gate equivalency of up to 1200/1800 gates
- 58/74 individually-configurable I/O pins allowing any mix of inputs, outputs or bidirectional signals (68/84-pin package)
- User -selectable TTL or HCMOS input threshold levels
- Multiple configuration modes for greatest flexibility and ease-of-use
- Verification feature allows user to check configuration data
- User-selectable security feature prevents read-back of configuration data
- Read-back of internal register states for system debug
- On-chip clock oscillator and clock buffer circuits provide flexible internal and external clocking functions
- Master reset of all internal register elements in addition to user-configurable Reset and/or Set control of individual CLB storage elements
- Complete development system support

General Description

The Logic Cell Array (LCA) is a high-density CMOS-integrated circuit available from Monolithic Memories. Its user-programmable array architecture is made up of three types of configurable elements: Input/Output Blocks, Logic Blocks and Interconnect. The designer can define individual I/O blocks for interface to external circuitry, define logic blocks to implement logic functions and define interconnection network to compose larger scale logic functions. The XACT™ Development system provides interactive graphic design capture and automatic routing. Both logic simulation and in-circuit emulation are available for design verification.

The Logic Cell Array is available in a variety of logic capacities, package styles, temperature ranges and speed grades.

PART NUMBER	LOGIC CAPACITY (USABLE GATES)	CONFIG-URABLE LOGIC BLOCKS	USER I/Os	CONFIG-URATION PROGRAM (BITS)
M2064	1200	64	58	12040
M2018	1800	100	74	17880

The Logic Cell Array's logic functions and interconnections are determined by data stored in internal static memory cells. On-chip logic provides for automatic loading of configuration data at power-up. The program data can reside in an EEPROM, EPROM or ROM on the circuit board or on a floppy disk or hard disk. The program can be loaded in a number of modes to accommodate various system requirements.

Package Availability

PART NUMBER	48-PIN PLASTIC DIP N48	68-PIN PLCC NL68	68-PIN PGA P68	84-PIN PLCC NL84	84-PIN PGA P84
M2064	X	X	X		
M2018		X	X	X	X

Ordering Information

M2018-70 C NL 84

- PART NUMBER
  - 2064 (1200 Gates, 58 IOB)
  - 2018 (1800 Gates, 74 IOB)
- SPEED GRADE
  - -33 (33 MHz Toggle Rate)
  - -50 (50 MHz Toggle Rate)
  - -70 (70 MHz Toggle Rate)
- NUMBER OF PINS
  - 48 (48 Pins)
  - 68 (68 Pins)
  - 84 (84 Pins)
- PACKAGE TYPE
  - NL = Pin Molded Chip Carrier
  - P = Pin Grid Array
  - N = Pin Molded DIP
- TEMPERATURE RANGE
  - C = Commercial
  - M = Military

2175 Mission College Blvd. Santa Clara, CA 95054-1592 Tel: (408) 970-9700 TWX: 910-338-2374
TWX: 910-338-2376
Monolithic Memories

Pin Description

$\overline{\text{PWRDWN}}$
An active low power-down input stops all internal activity to minimize VCC power and puts all output buffers in a high-impedance state. Configuration is retained, however, internal storage elements are Reset. When the $\overline{\text{PWRDWN}}$ pin returns HIGH, the device returns to operation with the same sequence of reset, buffer enable and DONE, $\overline{\text{PROGRAM}}$ as at the completion of configuration.

M0, RTRIG
As Mode 0, this input and M1, M2 are sampled before the start of configuration to establish the configuration mode to be used.

As a read trigger, an input transition to a HIGH, after configuration is complete, will initiate a readback of configuration and storage element data. This operation may be limited to a single request, or be inhibited altogether, by selecting the appropriate readback option when generating the bit stream.

M1, $\overline{\text{RDATA}}$
As Mode 1, this input and M0, M2 are sampled before the start of configuration to establish the configuration mode to be used.

As an active-low read data; after configuration is complete, this pin is the output of the readback data.

M2
As Mode 2, this input and M0, M1 are sampled before the start of configuration to establish the configuration, mode to be used. After configuration, this pin becomes a user-programmable I/O.

HDC
High during configuration is held at a HIGH level by the LCA until after configuration. It is intended to be available as a control indication that configuration is not complete. After configuration, this pin is a user I/O.

$\overline{\text{LDC}}$
Low during configuration is held at a LOW level by the LCA until after configuration. It is intended to be available as a control indication that configuration is not completed. It is particularly useful in master mode as a LOW enable for an EPROM. After configuration, this pin is a user I/O. If used as a LOW EPROM enable, it should be programmed as a HIGH after configuration.

$\overline{\text{RESET}}$
This is an active-low input which has three functions. Prior to the start of configuration, a LOW input will delay the start of the configuration process. An internal circuit senses the application of power and begins a minimal time-out cycle on the order of 100 ms. When the time-out and $\overline{\text{RESET}}$ are complete, the levels of the "M" mode lines are sampled and configuration begins. If $\overline{\text{RESET}}$ is asserted during a configuration, the LCA is reinitialized and will restart the configuration at the termination of RESET. If $\overline{\text{RESET}}$ is asserted after configuration is complete, it will provide an asynchronous reset of all IOB and CLB storage elements of the LCA.

DONE, $\overline{\text{PROG}}$
The DONE open drain output is configurable with or without a pull-up resistor of about 3 KΩ. At the completion of configuration, the circuitry of the LCA becomes active in a synchronous order and one configuration clock cycle later DONE is asserted. Once configuration is done, a HIGH-to-LOW transition of this program pin will cause an initialization of the LCA and start a reconfiguration if that mode is selected in the current configuration.

XTL1
This user I/O pin may be configured to operate as the output of an amplifier usable with an external crystal and bias circuitry to form an oscillator.

XTL2
This user I/O pin may be configured to operate as the input of an amplifier usable with an external crystal and bias circuitry to form an oscillator.

CCLK
During configuration, configuration clock is an output of an LCA in either master or peripheral mode. LCAs in slave mode use it as a clock input. During a readback operation, it is an input clock for the configuration data being output.

DOUT
This user I/O pin is used during configuration to output serial configuration data out for daisy-chained slaves' data in.

DIN
This user I/O pin is used as serial data in during slave or peripheral configuration. This pin is D0 in master configuration mode.

$\overline{\text{CSO}}$, $\overline{\text{CS1}}$, CS2, $\overline{\text{WRT}}$
These four inputs represent a set of signals, three active low and one active high, which are used in the peripheral mode to control configuration data entry. The assertion of all four generates a LOW CCLK and shifts DOUT data. The removal of any assertion clocks in the DIN data present and causes a HIGH CCLK. In master mode, these pins become part of the parallel configuration byte (D4, D3, D2, D1). After configuration is complete, they are user-programmed I/O.

$\overline{\text{RCLK}}$
During master mode configuration, this pin represents a read clock of an external memory device. After configuration is complete, this pin becomes a user-programmed I/O.

D0-D7
This set of eight pins represents the parallel configuration data byte for the master mode. After configuration is complete, they are user-programmed I/O.

A0-A15
This set of sixteen pins presents an address output for an external configuration memory during master mode. After configuration is complete, they are user-programmed I/O. A12 through A15 are not available in packages with less than sixty-eight pins.

I/O
A pin which may be programmed by the user to be input and/or output following configuration. Some of these pins present a high-impedance pull-up or perform other functions before configuration is complete.

Functional Description

The general structure of a Logic Cell Array is shown in Figure 1. The elements of the array include three categories of user-programmable elements: I/O Blocks, Configurable Logic Blocks and Programmable Interconnections. The I/O Blocks provide an interface between the logic array and the device package pins. The Configurable Logic Blocks perform user-specified logic functions, and the interconnect resources are programmed to form networks that carry logic signals among blocks.

Configuration of the Logic Cell Array is established through a distributed array of memory cells. The XACT development system generates the program used to configure the Logic Cell Array. The Logic Cell Array includes logic to implement automatic configuration.

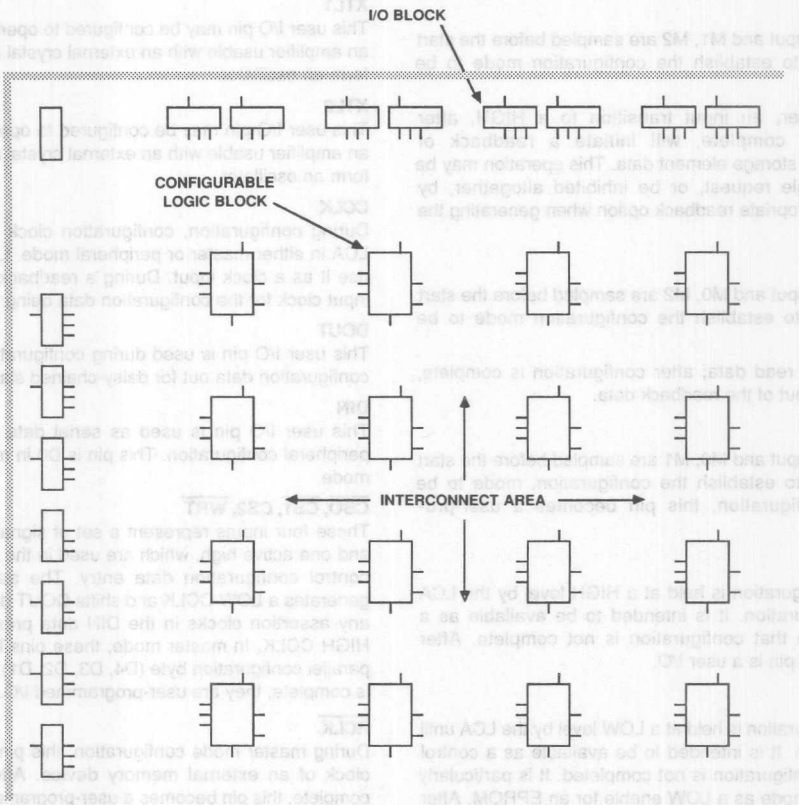


Figure 1. Logic Cell Array Structure

Configuration Memory

The configuration of the Monolithic Memories' Logic Cell Array is established by programming memory cells which determine the logic functions and interconnections. The memory loading process is independent of the user logic functions.

The static memory cell used for the configuration memory in the Logic Cell Array has been designed specifically for high reliability and noise immunity. Based on this design, integrity of the LCA configuration memory is assured even under adverse conditions. Compared with other programming alternatives, static memory provides the best combination of high density, high performance, high reliability and comprehensive testability. As shown in Figure 2, the basic memory cell consists of two CMOS inverters plus a pass transistor used for writing data to the cell. The cell is only written during configuration and only read during readback. During normal operation the pass transistor is "off" and does not affect the stability of the cell. This is quite different from the normal operation of conventional memory devices, in which the cells are continuously read and written.

The outputs Q and $\overline{Q}$ control pass-transistor gates directly. The absence of sense amplifiers and the output capacitive load provide additional stability to the cell. Due to the structure of the configuration memory cells, they are not affected by extreme power supply excursions or very high levels of alpha particle radiation. In reliability testing no soft errors have been observed, even in the presence of very high doses of alpha radiation.

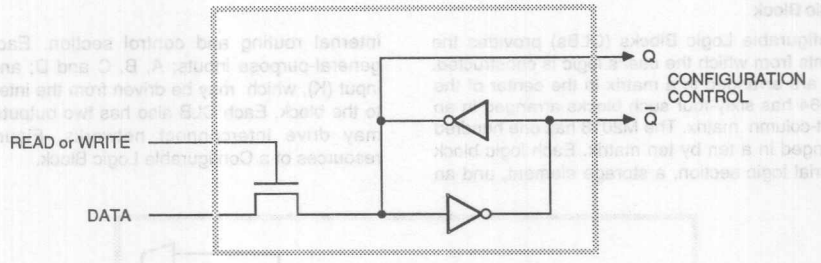


Figure 2. Configuration Memory Cell

Input/Output Block

Each user-configurable I/O block (IOB) provides an interface between the external package pin of the device and the internal logic. Each I/O block includes a programmable input path and a programmable output buffer. It also provides input clamping diodes to provide protection from electrostatic damage, and circuits to protect the LCA from latch-up due to input currents. Figure 3 shows the general structure of the I/O block.

The input buffer portion of each I/O block provides threshold detection to translate external signals applied to the package pin to internal logic levels. The input buffer threshold of the I/O blocks can be programmed to be compatible with either TTL (1.4 V) or CMOS (2.2 V) levels. The buffered input signal drives both the data input of an edge-triggered D-type flip-flop and one input of a two-input multiplexer. The output of the flip-flop provides the other input to the multiplexer. The user can select either the direct input path or the registered input, based on the content of the memory cell controlling the multiplexer. The I/O blocks along each edge of the die share common clocks. The flip-flops are reset during configuration as well as by the active-low chip $\overline{\text{RESET}}$ input.

Output buffers in the I/O blocks provide 4-mA drive for high fan-out CMOS or TTL-compatible signal levels. The output data (driving I/O block pin O) is the data source for the I/O block output buffer. Each I/O block output buffer is controlled by the contents of two configuration memory cells which turn the buffer ON or OFF or select logical three-state buffer control. The user may also select the output buffer three-state control (I/O block pin TS). When this I/O block output control signal is HIGH (a logic "1") the buffer is disabled and the package pin is high-impedance.

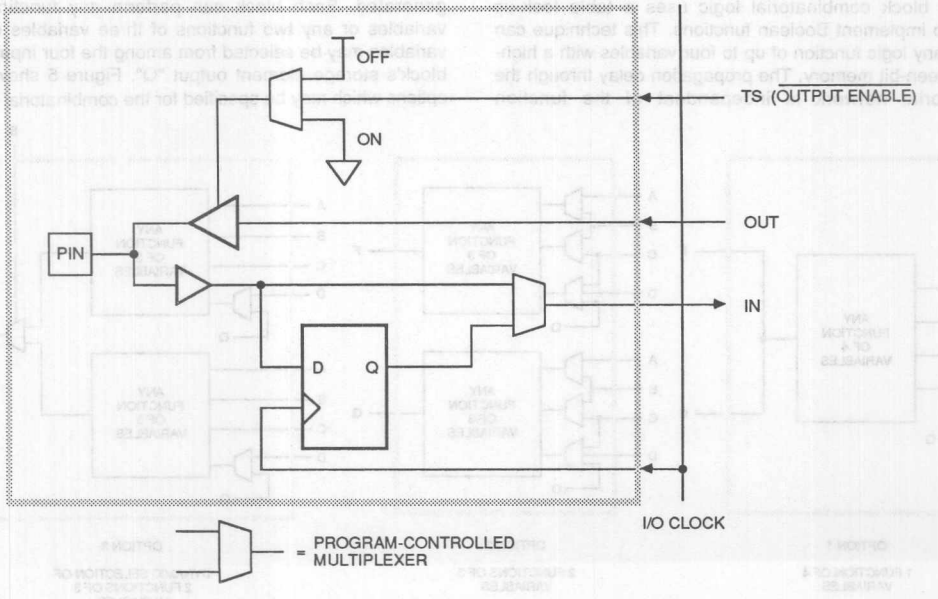


Figure 3. I/O Block

Configurable Logic Block

An array of Configurable Logic Blocks (CLBs) provides the functional elements from which the user's logic is constructed. The logic blocks are arranged in a matrix in the center of the device. The M2064 has sixty-four such blocks arranged in an eight-row by eight-column matrix. The M2018 has one hundred logic blocks arranged in a ten by ten matrix. Each logic block has a combinatorial logic section, a storage element, and an internal routing and control section. Each CLB has four general-purpose inputs; A, B, C and D; and a special clock input (K), which may be driven from the interconnect adjacent to the block. Each CLB also has two outputs, X and Y, which may drive interconnect networks. Figure 4 shows the resources of a Configurable Logic Block.

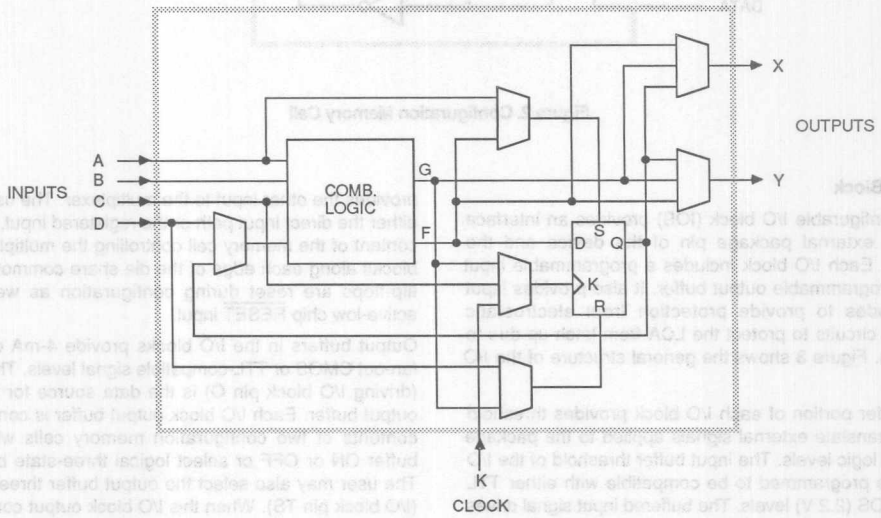


Figure 4. Configurable Logic Block

The logic block combinatorial logic uses a table look-up memory to implement Boolean functions. This technique can generate any logic function of up to four variables with a high-speed sixteen-bit memory. The propagation delay through the combinatorial network is independent of the function generated. Each block can perform any function of four variables or any two functions of three variables each. The variables may be selected from among the four inputs and the block's storage element output "Q". Figure 5 shows various options which may be specified for the combinatorial logic.

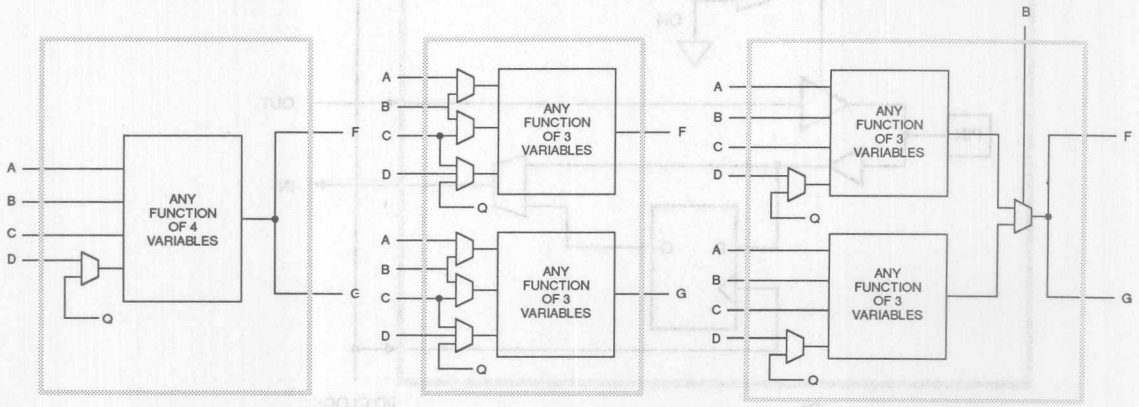


Figure 5. CLB Combinatorial Logic Options

If the single four-variable configuration is selected (Option 1), the F and G outputs are identical. If the two-function alternative is selected (Option 2), logic functions F and G may be independent functions of three variables each. The three variables can be selected from among the four logic block inputs and its storage element output Q. A third form of the combinatorial logic (Option 3) is a special case of the two-function form in which the B input dynamically selects between the two function tables providing a single merged logic function output. This dynamic selection allows some five-variable functions to be generated from the four block inputs and storage element Q. Combinatorial functions are restricted in that one may not use both its storage element output Q and the input variable of the logic block pin D in the same function.

If used, the storage element in each Configurable Logic Block (Figure 6) can be programmed to be either an edge-sensitive D-type flip-flop or a level-sensitive D latch. The clock or enable for each storage element can be selected from:

- The special-purpose clock input K
- The general-purpose input C
- The combinatorial function G

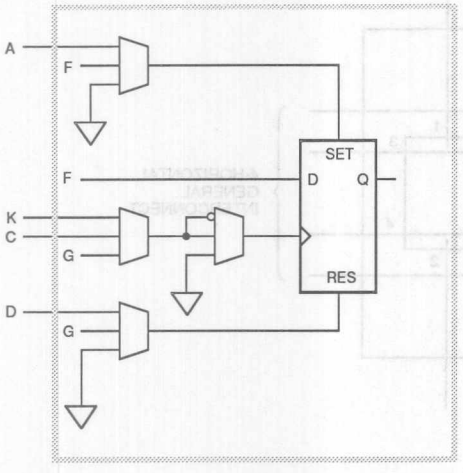


Figure 6. CLB Storage Element

The user may also select the clock active sense within each logic block. This programmable inversion eliminates the need to route both phases of a clock signal throughout the device.

The storage element data input is supplied from the function F output of the combinatorial logic. Asynchronous SET and RESET controls are provided for each storage element. The user may enable these controls independently and select their source. They are active-high inputs and the asynchronous reset is dominant. The storage elements are reset by the active-low chip $\overline{\text{RESET}}$ pin as well as by the initialization phase preceding configuration. If the storage element is not used, it is disabled.

The two block outputs, X and Y, can be driven by either the combinatorial functions, F or G, or the storage element output Q (Figure 4). Selection of the outputs is completely interchangeable and may be made to optimize routing efficiencies of the networks interconnecting the logic blocks and I/O blocks.

Programmable Interconnect

Programmable interconection resources in the Logic Cell Array provide routing paths to connect inputs and outputs of the I/O and logic blocks into desired networks. All interconnections are composed of metal segments, with programmable switching points provided to implement the necessary routing. Three types of resources accommodate different types of networks:

- General purpose interconnect
- Long lines
- Direct connection

General-Purpose Interconnect

General-purpose interconnect, as shown in Figure 7a, is composed of four horizontal metal segments between the rows and five vertical metal segments between the columns of logic and I/O blocks. Each segment is only the "height" or "width" of a logic block. Where these segments would cross at the intersections of rows and columns, switching matrices are provided to allow interconnections of metal segments from the adjoining rows and columns. Switches in the switch matrices and on block outputs are specially designed transistors, each controlled by a configuration bit.

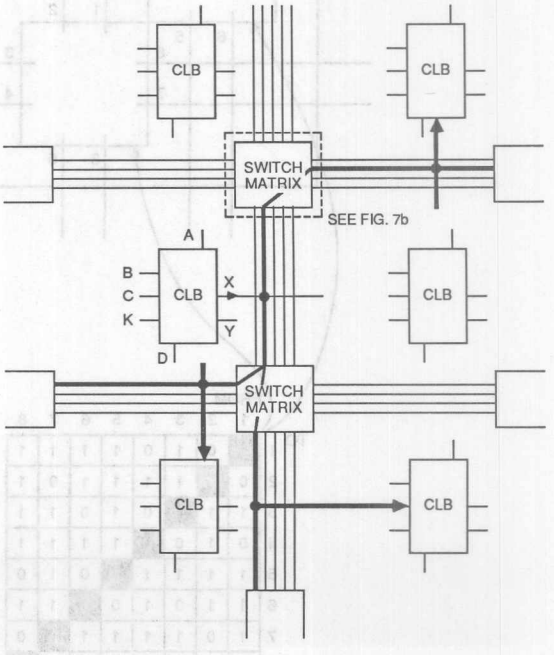
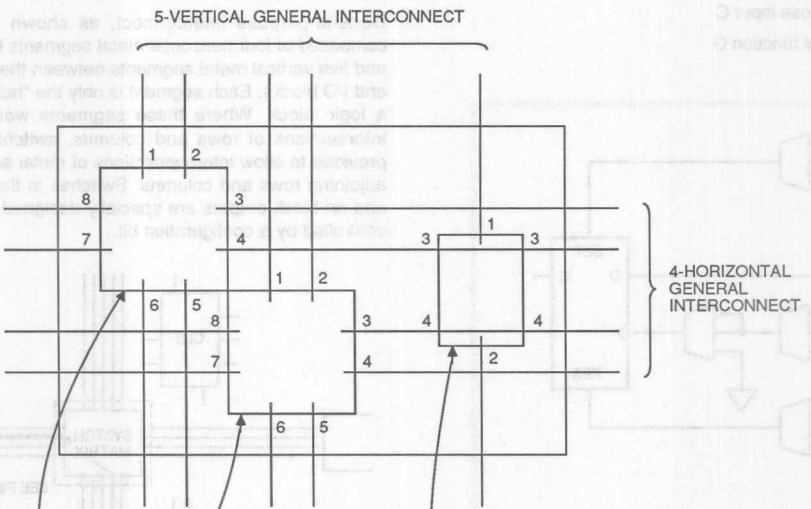


Figure 7a. General-Purpose Interconnect

Logic block output switches provide contacts to adjacent general interconnect segments and therefore to the switching matrix at each end of those segments. A switch matrix can connect an interconnect segment to other segments to form a network. Figure 7a shows the general interconnect used to route a signal from one logic block to three other logic blocks. As shown, combinations of closed switches in a switch matrix allow multiple branches for each network. The inputs of the logic or I/O blocks are multiplexers that can be programmed with configuration bits to select an input network from the adjacent interconnect segments. Since the switch connections to block inputs are undirectional (as are block outputs) they are usable *only* for input connections. The development system software provides automatic routing of these interconnections. Interactive routing is also available for design optimization. This is accomplished by selecting a network and then toggling the states of the interconnect points by selecting them with the "mouse". In this mode, the connections through the switch matrix may be established by selecting pairs of matrix pins. The switching matrix combinations are indicated in Figure 7b.

Special buffers within the interconnect area provide periodic signal isolation and restoration for higher general interconnect fan-out and better performance. The repowering buffers are bidirectional, since signals must be able to propagate in either direction on a general interconnect segment. Direction controls are automatically established by the Logic Cell Array development system software. Repowering buffers are provided only for the general-purpose interconnect since the direct and long-line resources do not exhibit the same R-C delay accumulation. The Logic Cell Array is divided into nine sections with buffers automatically provided for general interconnect at the boundaries of these sections. These boundaries can be viewed with the development system. For routing within a section, no buffers are used. The delay calculator of the XACT development system automatically calculates and displays the block, interconnect and buffer delays for any selected paths.



FROM / TO

TO \ FROM	1	2	3	4	5	6	7	8
1	■	0	1	0	1	1	1	1
2	0	■	1	1	1	1	0	1
3	1	1	■	0	1	0	1	1
4	0	1	0	■	1	1	1	1
5	1	1	1	1	■	0	1	0
6	1	1	0	1	0	■	1	1
7	1	0	1	1	1	1	■	0
8	1	1	1	1	0	1	0	■

FROM / TO

TO \ FROM	1	2	3	4
1	■	1	1	1
2	1	■	1	1
3	1	1	■	0
4	1	1	0	■

1 = VALID CONNECTION
0 = INVALID CONNECTION

Figure 7b. Interconnection Switching Matrix

Long Lines

Long lines, shown in Figure 8a, run both vertically and horizontally the height or width of the interconnect area. Each vertical interconnection column has two long lines; each horizontal row has one, with an additional long line adjacent to each set of I/O blocks. The long lines bypass the switch matrices and are intended primarily for signals that must travel a long distance or must have minimum skew among multiple destinations.

A global buffer in the Logic Cell Array is available to drive a single signal to all B and K inputs of logic blocks. Using the global buffer for a clock provides a very low skew, high fan-out synchronized clock for use at any or all of the logic blocks. At each block, a configuration bit for the K input to the block can select this global line as the storage element clock signal. Alternatively, other clock sources can be used.

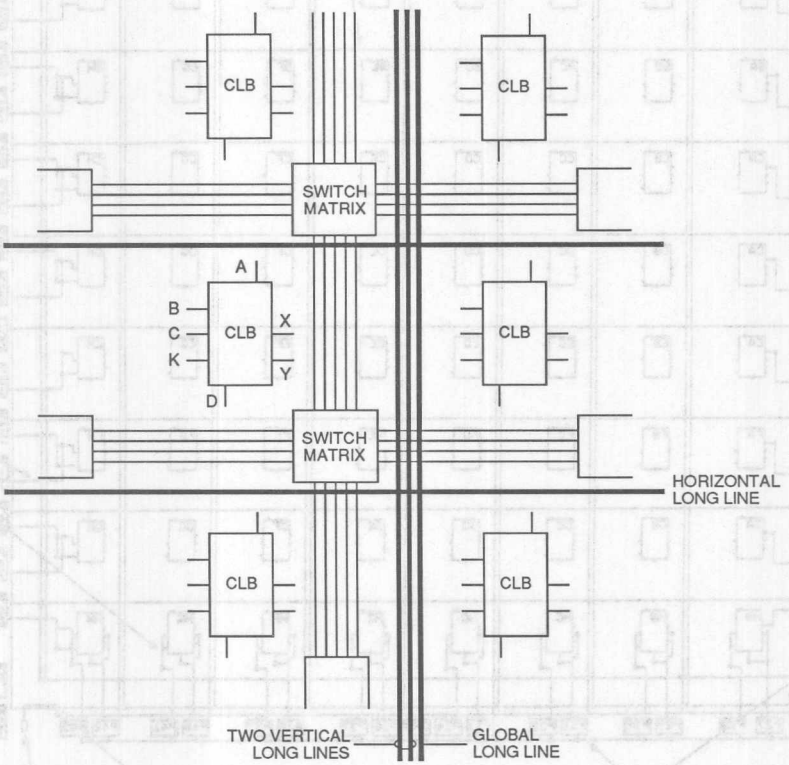


Figure 8a. Long Line Interconnect

A second buffer below the bottom row of the array drives a horizontal long line which, in turn, can drive a vertical long line in each interconnection column. This alternate buffer also has low skew and high fan-out capability. The network formed by this alternate buffer's long lines can be selected to drive the B, C or K inputs of the logic blocks. Alternatively, these long lines can be driven by a logic or I/O block on a column-by-column basis. This capability provides a common, low-skew clock or control line within each column of logic blocks. Interconnections of these long lines are shown in Figure 8b.

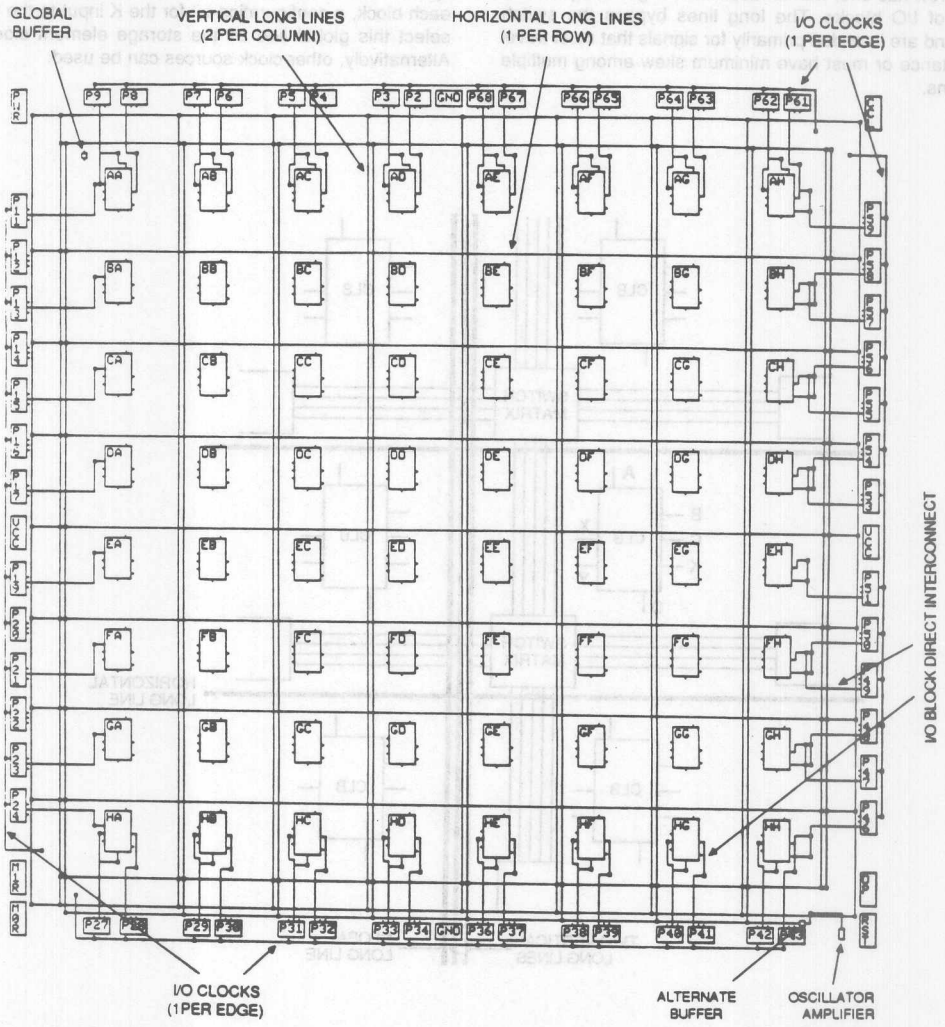


Figure 8b. M2064 Long Lines, I/O Clocks, I/O Direct Interconnect

Direct Interconnect

Direct interconnect, shown in Figure 9, provides the most efficient implementation of networks between adjacent logic or I/O blocks. Signals routed from block to block by means of direct interconnect exhibit minimum interconnect propagation and use minimum interconnect resources. For each CLB, the X output may be connected directly to the C or D inputs of the CLB above and to the A or B inputs of the CLB below it. The Y output can use direct interconnect to drive the B input of the block immediately to its right. Where logic blocks are adjacent to I/O blocks, direct connect is provided to the I/O block input (I) on the left edge of the die, the output (O) on the right edge, or both on I/O blocks at the top and bottom of the die. Direct interconnections of I/O blocks with CLBs are shown in Figure 8b.

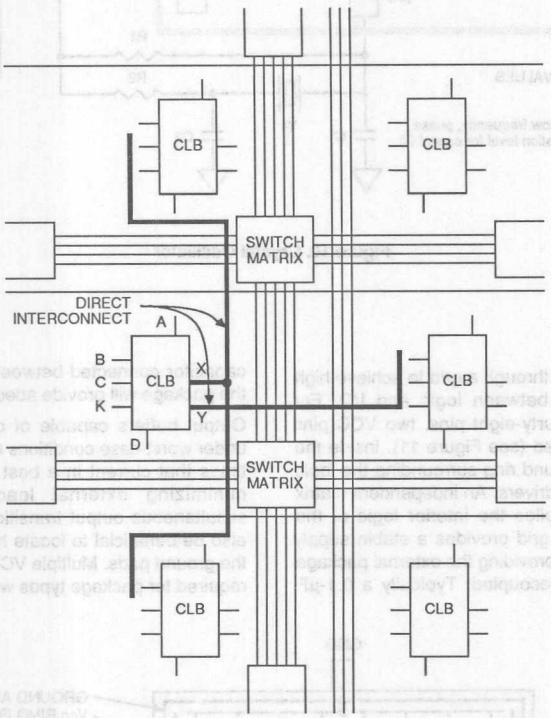
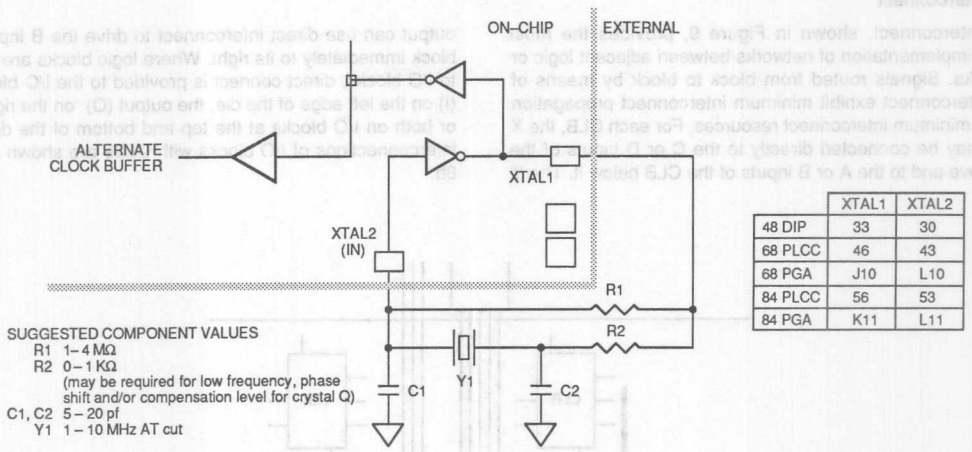


Figure 9. Direct Interconnect

Crystal Oscillator

An internal high-speed inverting amplifier is available to implement an on-chip crystal oscillator. It is associated with the auxiliary clock buffer in the lower right corner of the die. When configured to drive the auxiliary clock buffer, two special adjacent user I/O blocks are also configured to connect the oscillator amplifier with external crystal oscillator components, as shown in Figure 10. This circuit becomes active before configuration is complete in order to allow the oscillator to stabilize. Actual internal connection is delayed until completion of configuration. The feedback resistor R1 between output and input, biases the amplifier at threshold. It should be as large a value as practical to minimize loading of the crystal. The inversion of the amplifier, together with the R-C networks and crystal, produces the 360-degree phase shift of the Pierce oscillator.

A series resistor R2 may be included to add to the amplifier output impedance when needed for phase-shift control or crystal resistance matching or to limit the amplifier input swing to control clipping at large amplitudes. Excess feedback voltage may be adjusted by the ratio of C2/C1. The amplifier is designed to be used over the range from 1 MHz up to one-half the specified CLB toggle frequency. Use at frequencies below 1 MHz may require individual characterization with respect to a series resistance. Operation at frequencies above 20 MHz generally requires a crystal to operate in a third overtone mode, in which the fundamental frequency must be suppressed by the R-C networks. When the amplifier does not drive the auxiliary buffer, these I/O blocks and their package pins are available for general user I/O.



	XTAL1	XTAL2
48 DIP	33	30
68 PLCC	46	43
68 PGA	J10	L10
84 PLCC	56	53
84 PGA	K11	L11

Figure 10. Crystal Oscillator

Power

Power Distribution

Power for the LCA is distributed through a grid to achieve high noise immunity and isolation between logic and I/O. For packages having more than fourty-eight pins, two VCC pins and two ground pins are provided (see Figure 11). Inside the LCA, a dedicated VCC and ground ring surrounding the logic array provides power to the I/O drivers. An independent matrix of VCC and ground lines supplies the interior logic of the device. This power distribution grid provides a stable supply and ground for all internal logic, providing the external package power pins are appropriately decoupled. Typically a 0.1-μF capacitor connected between the VCC and ground pins near the package will provide adequate decoupling.

Output buffers capable of driving the specified 4-mA loads under worst-case conditions may be capable of driving 25 to 30 times that current in a best case. Noise can be reduced by minimizing external load capacitance and reducing simultaneous output transitions in the same direction. It may also be beneficial to locate heavily-loaded output buffers near the ground pads. Multiple VCC and ground pin connections are required for package types which provide them.

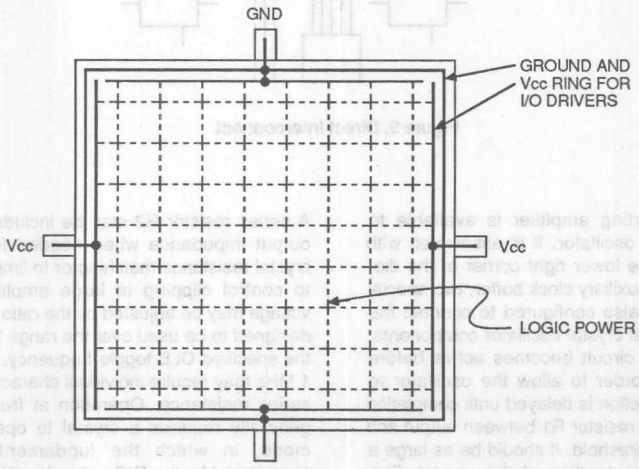


Figure 11. LCA Power Distribution

Power Dissipation

The Logic Cell Array exhibits the low power consumption characteristic of CMOS ICs. Only quiescent power is required for the LCA configured for CMOS input levels. The TTL input level configuration option requires additional power for level shifting. The power required by the static memory cells which hold the configuration data is very low and may be maintained in a power-down mode.

Typically most of power dissipation is produced by capacitive loads on the output buffers, since the power per output is 25 µW/pF/MHz. Another component of I/O power is the DC loading on each output pin. For any given system, the user can calculate the power requirement based on the resistive loading of the devices driven by the Logic Cell Array.

Internal power supply dissipation is a function of clock frequency and the number of nodes changing on each clock. In an LCA the fraction of nodes changing on a given clock is typically low (10-20%). For example, in a 16-bit binary counter, the average clock produces a change in slightly less than two of the sixteen bits. In a 4-input AND gate there will be two transitions in sixteen states. Typical global clock buffer power is about 3 mW/MHz for the M2064 and 4 mW/MHz for the M2018. With a "typical" load of three general interconnect segments, each CLB output requires about 0.4 mW/MHz of its output frequency. Graphs of power versus operating frequency are shown in Table 1.

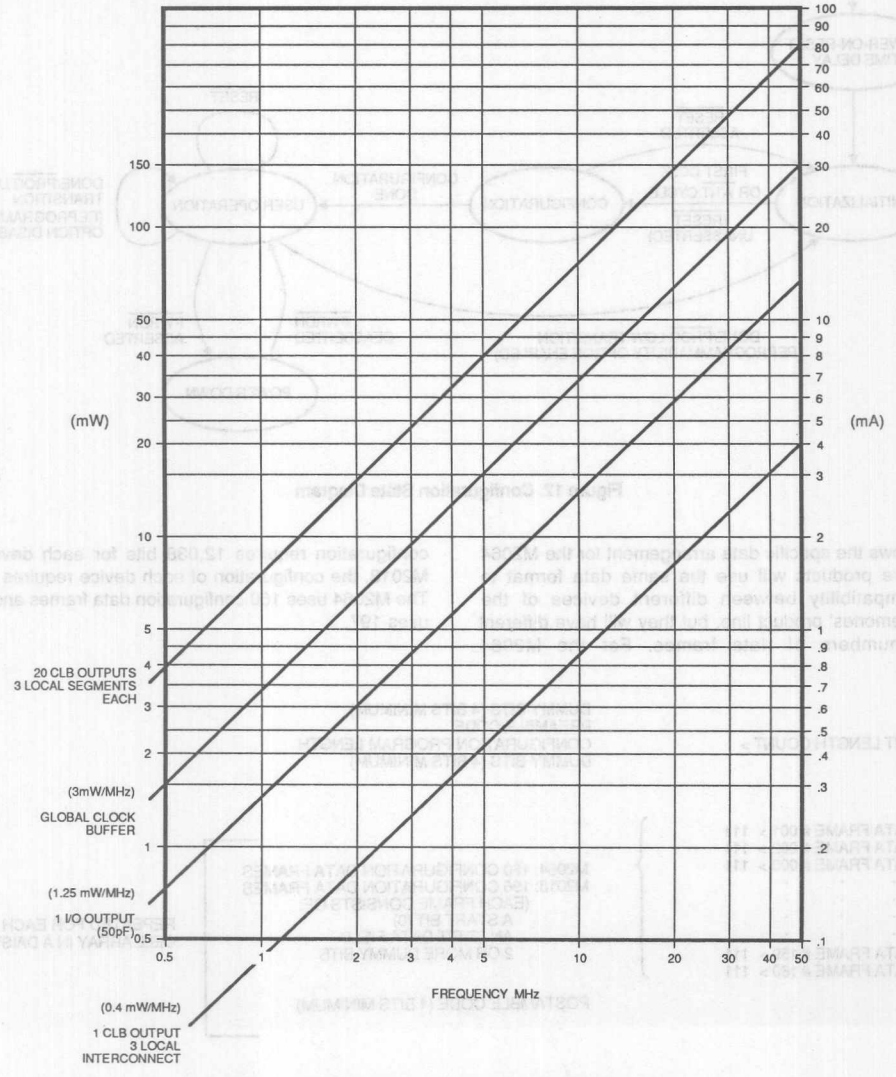


Table 1. Typical LCA Power Consumption by Element

Programming

Configuration data to define the function and interconnection within a Logic Cell Array are loaded automatically at power-up or upon command. Several methods of automatically loading the required data are designed into the Logic Cell Array and are determined by logic levels applied to mode selection pins at configuration time. The form of the data may be either serial or parallel, depending on the configuration mode. The programming data are independent of the configuration mode selected. The state diagram of Figure 12 illustrates the configuration process.

Input thresholds for user I/O pins can be selected to be either TTL-compatible or CMOS-compatible and remain in that state until the LCA begins operation. If the user has selected CMOS compatibility, the input thresholds are changed to CMOS levels during configuration.

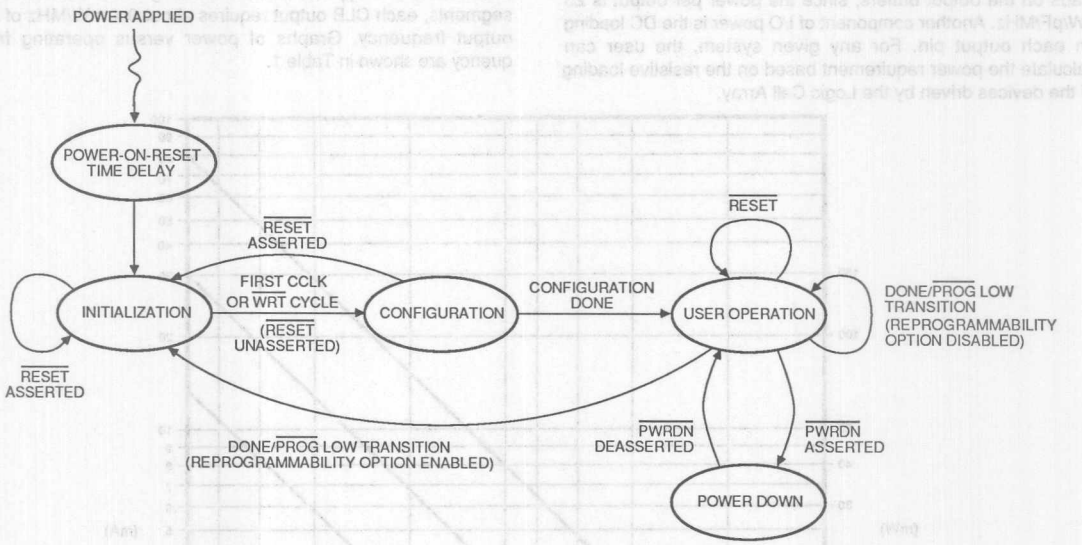


Figure 12. Configuration State Diagram

Figure 13 shows the specific data arrangement for the M2064 device. Future products will use the same data format to maintain compatibility between different devices of the Monolithic Memories' product line, but they will have different sizes and numbers of data frames. For the M2064 configuration requires 12,038 bits for each device. For the M2018, the configuration of each device requires 17,878 bits. The M2064 uses 160 configuration data frames and the M2018 uses 197.

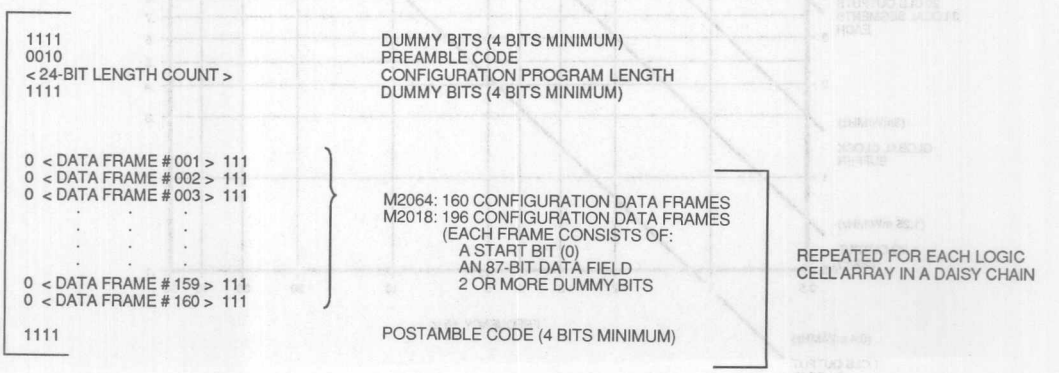


Figure 13. M2064 Configuration Data Arrangement

The configuration bit stream begins with preamble bits, a preamble code and a length count. The length count is loaded into the control logic of the Logic Cell Array and is used to determine the completion of the configuration process. When configuration is initiated, a 24-bit length counter is set to 0 and begins to count the total number of configuration clock cycles applied to the device. When the current length count equals the loaded length count, the configuration process is complete. Two clocks before completion, the internal logic becomes active and is reset. On the next clock, the inputs and outputs become active as configured and consideration should be given to avoid configuration signal contention. (Attention must be paid to avoid contention on pins which are used as inputs during configuration and become outputs in operation.) On the last configuration clock, the completion of configuration is signalled by the release of the DONE, $\overline{PROG}$ pin of the device as the device begins operation. This open-drain output can be AND-tied with multiple Logic Cell Arrays and used as an active-high READY or active-low, $\overline{RESET}$, to other portions of the system. High during configuration (HDC) and low during configuration ($\overline{LDC}$), are released one CCLK cycle before DONE is asserted. In master mode configurations, it is convenient to use $\overline{LDC}$ as an active-low EPROM chip enable.

As each data bit is supplied to the LCA, it is internally assembled into a data word. As each data word is completely assembled, it is loaded in parallel into one word of the internal configuration memory array. The last word must be loaded before the current length count compare is true. If the configuration data are in error, e.g., PROM address lines swapped, the LCA will not be ready at the length count and the counter will cycle through an additonal complete count prior to configuration being "done".

Figure 14 shows the selection of the configuration mode based on the state of the mode pins M0 and M1. These package pins are sampled prior to the start of the configuration process to determine the mode to be used. Once configuration is DONE and subsequent operation has begun, the mode pins may be used to perform data readback, as discussed later. An additional mode pin, M2, must be defined at the start of configuration. This package pin is a user-configurable I/O after configuration is complete.

MODE PIN			MODE SELECTED
M0	M1	M2	
0	0	0	Master serial
0	0	1	Master LOW mode
0	1	1	Master HIGH mode
1	0	1	Peripheral mode
1	1	1	Slave mode

Master LOW addresses begin at 0000 and increment.
Master HIGH addresses begin at FFFF and decrement.

Figure 14. Configuration Mode Selection

Initialization Phase

When power is applied, an internal power-on-reset circuit is triggered. When VCC reaches the voltage at which the LCA begins to operate (2.5 to 3 Volts), the chip is initialized, outputs are made high-impedance and a time-out is initiated to allow time for power to stabilize. This time-out (15 to 35 ms) is determined by a counter driven by a self-generated, internal sampling clock that drives the configuration clock (CCLK) in master configuration mode. This internal sampling clock will vary with process, temperature and power supply over the range of 0.5 to 1.5 MHz. LCAs with mode lines set for master mode will time-out of their initialization using a longer counter (60 to 140 ms) to assure that all devices, which it may be driving in a daisy chain, will be ready. Configuration using peripheral or slave modes must be delayed long enough for this initialization to be completed.

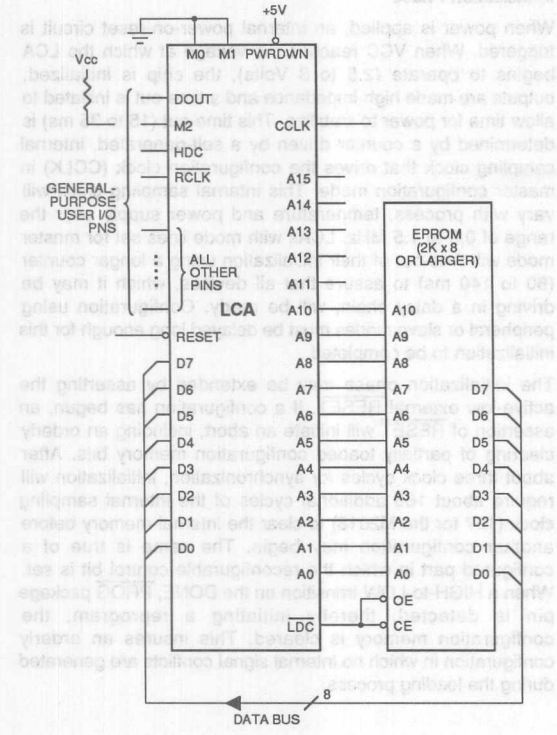
The initialization phase may be extended by asserting the active-low external $\overline{RESET}$. If a configuration has begun, an assertion of $\overline{RESET}$ will initiate an abort, including an orderly clearing of partially loaded configuration memory bits. After about three clock cycles for synchronization, initialization will require about 160 additional cycles of the internal sampling clock (197 for the M2018) to clear the internal memory before another configuration may begin. The same is true of a configured part in which the reconfigurable control bit is set. When a HIGH-to-LOW transition on the DONE, $\overline{PROG}$ package pin is detected, thereby initiating a reprogram, the configuration memory is cleared. This insures an orderly configuration in which no internal signal conflicts are generated during the loading process.

Master Mode

In master mode, the Logic Cell Array automatically loads the configuration program from an external memory device. Figure 15a shows an example of the master mode connections required. The Logic Cell Array provides sixteen address outputs and the control signals $\overline{RCLK}$ (read clock), HDC (high during configuration) and $\overline{LDC}$ (low during configuration) to execute read cycles from the external memory. Parallel eight-bit data words are read and internally serialized. As each data word is read, the least significant bit of each byte, normally D0, is the next bit in the serial stream.

Addresses supplied by the Logic Cell Array can be selected by the mode lines to begin at address 0 and incremented to read the memory (master low mode), or they can begin at address FFFF Hex and be decremented (master high mode). This capability is provided to allow the Logic Cell Array to share external memory with another device, such as a microprocessor. For example, if the processor begins its execution from low memory, the Logic Cell Array can load itself from high memory and enable the processor to begin execution once configuration is completed. The DONE, $\overline{PROG}$ output pin can be used to hold the processor in a Reset state until the Logic Cell Array has completed the configuration process.

The master serial mode uses serial configuration data, synchronized by the rising edge of RCLK, as in Figure 15b.



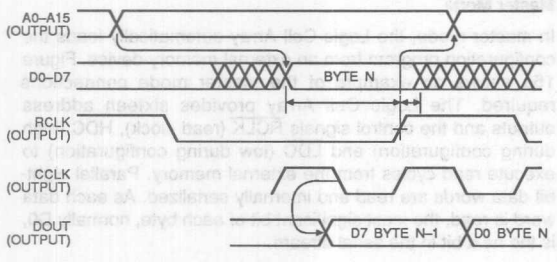


Figure 15a. Master Low Address Configuration

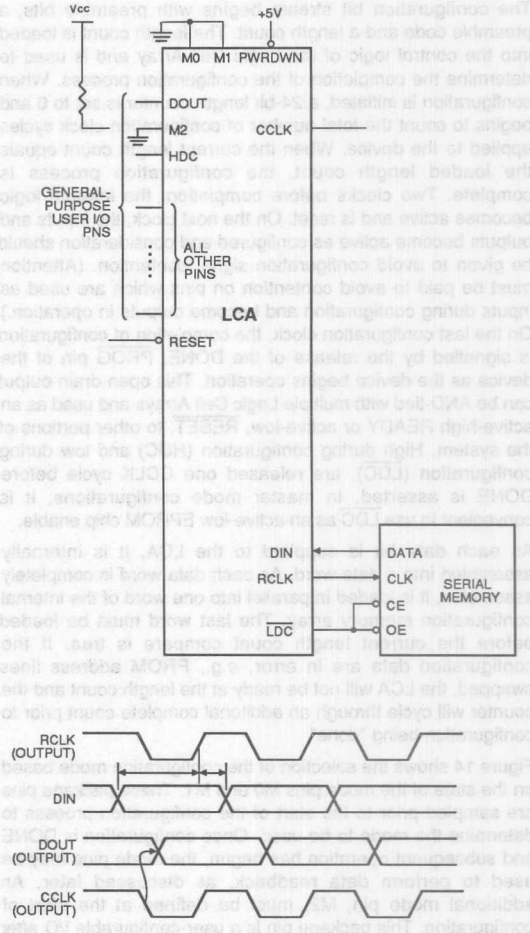


Figure 15b. Master Serial Mode Configuration

Peripheral Mode

Peripheral mode provides a simplified interface through which the device may be loaded as a processor peripheral. Figure 16 shows the peripheral mode connections. Processor write cycles are decoded from the common assertion of the active-low write strobe ($\overline{WRT}$), and two active-low and one active-high chip selects ($\overline{CS0}$, $\overline{CS1}$, CS2). If all these signals are not available, the unused inputs should be driven to their respective active levels. The Logic Cell Array will accept one bit of the configuration program on the data input (DIN) pin for each processor write cycle. Data is supplied in the serial sequence described earlier.

Since only a single bit from the processor data bus is loaded per cycle, the loading process involves the processor reading a byte or word of data, writing a bit of the data to the Logic Cell Array, shifting the word and writing a bit until all bits of the word are written, then continuing in the same fashion with the next word, etc. After the configuration program has been loaded, an additional three clocks (a total of three more than the length count) must be supplied in order to complete the configuration process. When more than one device is being used in the system, each device can be assigned a different bit in the processor data bus, and multiple devices can be loaded on each processor write cycle. This "broadside" loading method provides a very easy and time-efficient method of loading several devices.

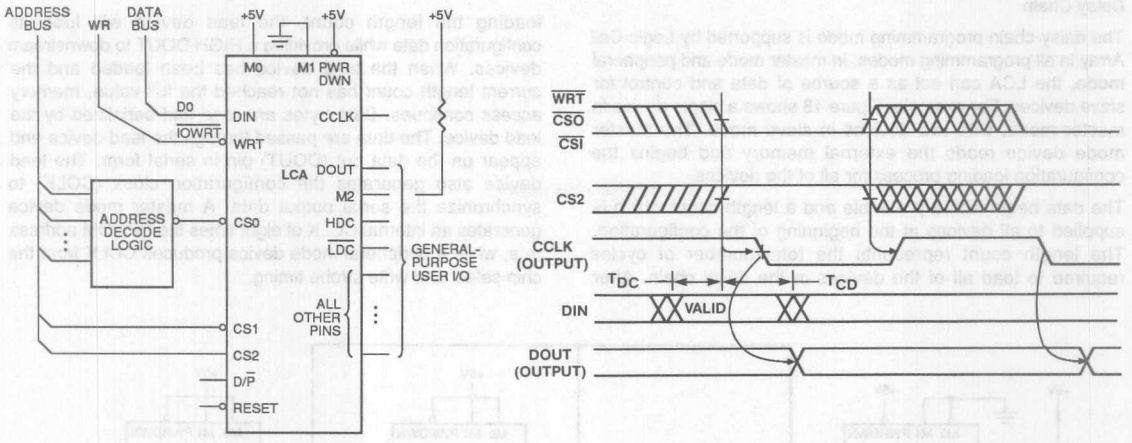


Figure 16. Peripheral Mode Configuration

2

Slave Mode

Slave mode, Figure 17, provides the simplest interface for loading the Logic Cell Array configuration. Data is supplied in conjunction with a synchronizing clock. For each LOW-to-HIGH input transition of configuration clock (CCLK), the data present on the data input (DIN) pin is loaded into the internal shift register. Data may be supplied by a processor or by other special circuits. Slave mode is used for downstream devices in a daisy-chain configuration. The data for each slave LCA are supplied by the preceding LCA in the chain, and the clock is supplied by the lead device, which is configured in master or peripheral mode. After the configuration program has been loaded, an additional three clocks (a total of three more than the length count) must be supplied in order to complete the configuration process.

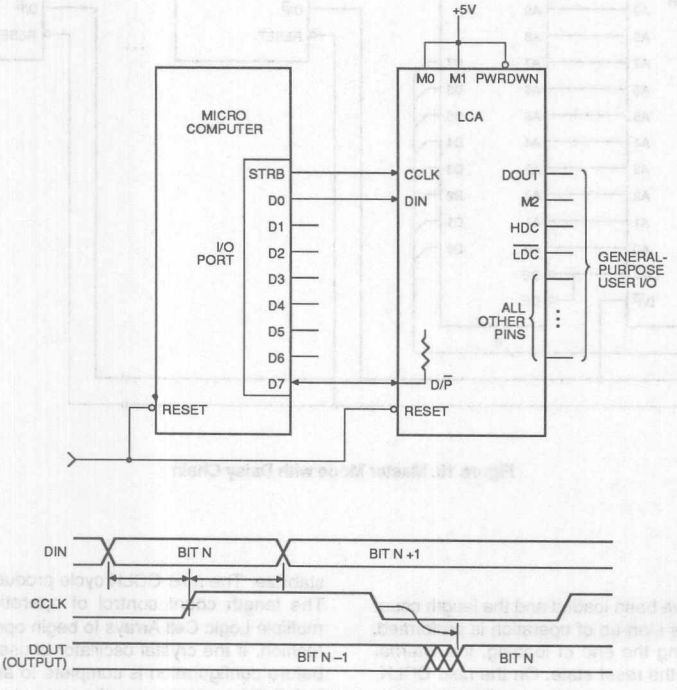


Figure 17. Slave Mode Configuration

Daisy Chain

The daisy-chain programming mode is supported by Logic Cell Array in all programming modes. In master mode and peripheral mode, the LCA can act as a source of data and control for slave devices. For example, Figure 18 shows a single device in master mode, with two devices in slave mode. The master mode device reads the external memory and begins the configuration loading process for all of the devices.

The data begin with a preamble and a length count which is supplied to all devices at the beginning of the configuration. The length count represents the total number of cycles required to load all of the devices in the daisy chain. After loading the length count, the lead device will load its configuration data while providing a HIGH DOUT to downstream devices. When the lead device has been loaded and the current length count has not reached the full value, memory access continues. Data bytes are read and serialized by the lead device. The data are passed through the lead device and appear on the data out (DOUT) pin in serial form. The lead device also generates the configuration clock (CCLK) to synchronize the serial output data. A master mode device generates an internal CCLK of eight times the EPROM address rate, while a peripheral mode device produces CCLK from the chip select and write strobe timing.

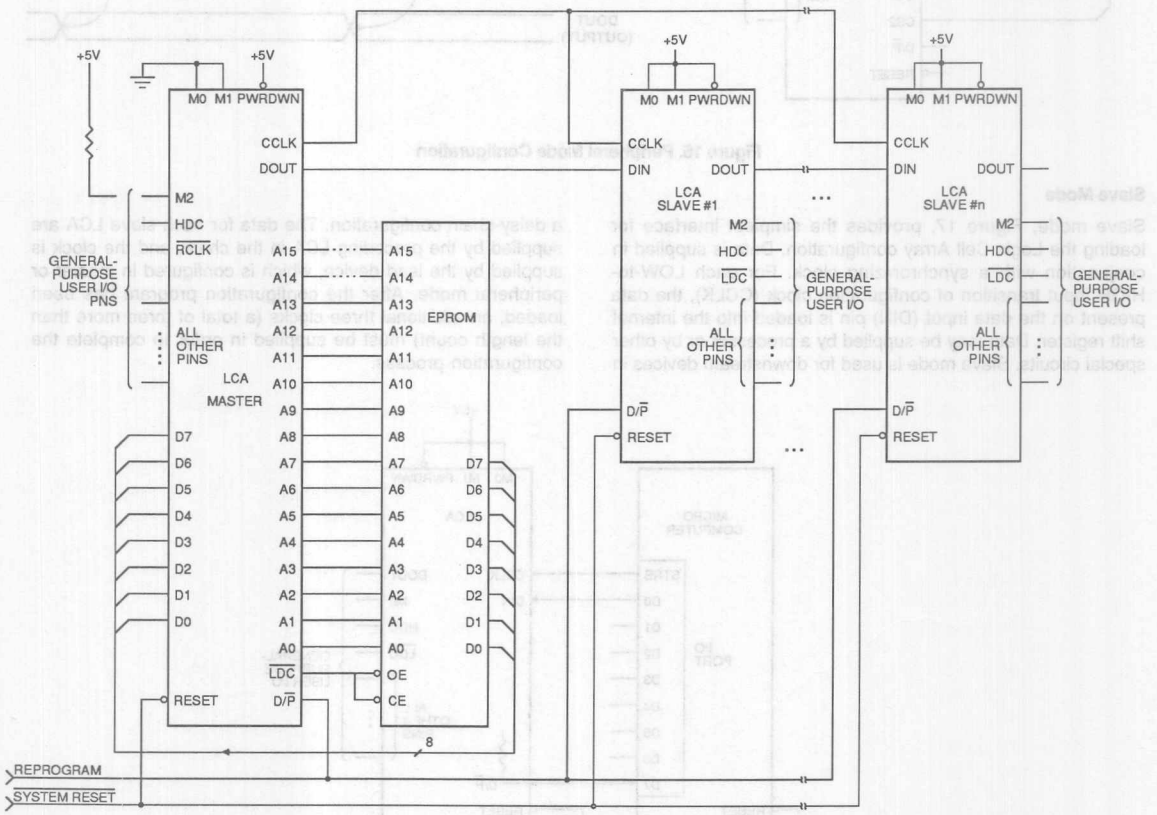


Figure 18. Master Mode with Daisy Chain

Operation

When all of the devices have been loaded and the length count is complete, a synchronous start-up of operation is performed. On the clock cycle following the end of loading, the internal logic begins functioning in the reset state. On the next CCLK, the configured output buffers become active to allow signals to stabilize. The next CCLK cycle produces the DONE condition. The length count control of operation allows a system of multiple Logic Cell Arrays to begin operation in a synchronized fashion. If the crystal oscillator is used, it will begin operation before configuration is complete to allow time for stabilization before it is connected to the internal circuitry.

Special Features

In addition to the normal user logic functions and interconnect, the configuration data include control for several special functions:

- Input thresholds
- Readback enable
- Reprogram enable
- DONE pull-up resistor

Each of these functions is controlled by a portion of the configuration program generated by the XACT Development System.

Input Thresholds

During configuration, all input thresholds are TTL level. During configuration input thresholds are established as specified, either TTL or CMOS. The $\overline{\text{PWRDN}}$ input threshold is an exception; it is always a CMOS level input. The TTL threshold option requires additional power for threshold shifting.

Readback

After a Logic Cell Array has been programmed, the configuration program may be read back from the device. Readback may be used for verification of configuration and as a method of determining the state of internal logic nodes during debugging. In applications in which the verification is not used, it may be desirable to limit access to the configuration data. Three readback options are provided: 'on command', 'only once', and 'never'. If 'on-command readback' is selected, the device will respond to all readback requests. If 'readback once' is selected, the device will respond only to the first readback request after programming is complete. Subsequent readback requests will be ignored. If 'readback never' is selected, the device will not respond to a readback command.

Readback is accomplished without the use of any of the user I/O pins; only M0, M1, and CCLK pins are used. An initiation of readback is produced by a LOW-to-HIGH transition of the M0, RTRIG (read trigger) pin. Once the readback command has been given, CCLK is cycled to read back each data bit in a format similar to loading. After two dummy bits, the first data frame is shifted out, in inverted sense, on the M1, $\overline{\text{RDATA}}$ (read data) pin. All data frames must be read back to complete the process and return the mode select and CCLK pins to their normal functions.

In addition to the configuration program, the readback includes the current state of each of the internal logic block storage elements, and the state of the input (I) connection pin on each I/O block. This state information is used by the Logic Cell Array development system In-Circuit Emulator to provide visibility into the internal operation of the logic while the system is operating. To readback a uniform time sample of all storage elements it may be necessary to inhibit the system clock.

Reprogram

The configuration memory of the Logic Cell Array may be rewritten while the device is in the user's system, if that option is selected when the LCA is configured. If another programming cycle is to be initiated, the dual function package pin DONE, $\overline{\text{PROG}}$ must be given a HIGH-to-LOW transition. Sensitivity to noise is reduced, by confirming the HIGH-to-LOW transition over two to three cycles using the LCA's internal sampling oscillator. When a reprogram command is recognized, all internal logic and connectivity definitions are erased and the I/O package pins are forced to a high impedance condition. The device returns to the initialization state. Reprogram control is often implemented with an external open collector driver which pulls DONE, $\overline{\text{PROG}}$ LOW. Once it recognizes a stable request, the Logic Cell Array will hold a LOW until the new configuration has been completed. Whether or not the reprogram request is maintained, the Logic Cell Array will begin operation upon completion of configuration.

DONE Pull-up

The DONE, $\overline{\text{PROG}}$ pin is an open drain I/O that indicates programming status. As an input, it initiates a reprogram operation. An optional internal pull-up resistor may be enabled.

Battery Backup

Because the control store of the Logic Cell Array is a CMOS static memory, its cells require only a very low standby current for data retention. In some systems, this low data retention current characteristic facilitates preserving configurations in the event of a primary power loss. The Logic Cell Array has built in power-down logic which, when activated, will disable normal operation of the device and retain only the configuration data. All internal operation is suspended and output buffers are placed in their high-impedance state.

Power-down data retention is possible with a simple battery-backup circuit because the power requirement is extremely low. For retention at 2.0 V, the required current is typically on the order of 0.5 mA. Screening of this parameter is available. To force the Logic Cell Array into the power-down state, the user must pull the $\overline{\text{PWRDWN}}$ pin low and continue to supply a retention voltage to the VCC pins of the package. When normal power is restored, VCC is elevated to its normal operating voltage and $\overline{\text{PWRDWN}}$ is returned to a HIGH. The Logic Cell Array resumes operation with the same internal sequence that occurs at the conclusion of configuration. Internal I/O and logic block storage elements will be reset, the outputs will become enabled and then the DONE, $\overline{\text{PROG}}$ pin will be released. No configuration programming is involved.

Performance

The high performance of the Logic Cell Array results from its patented architectural features and from the use of an advanced high-speed CMOS manufacturing process. Performance may be measured in terms of minimum propagation times for logic elements.

Flip-flop loop delays for the I/O block and logic block flip-flops are about 3 ns. This short delay provides very good performance under asynchronous clock and data conditions. Short loop delays minimize the probability of a metastable condition which can result from assertion of the clock during data transitions. Because of the short loop delay characteristic in the Logic Cell Array, the I/O block flip-flops can be used very effectively to synchronize external signals applied to the device. Once synchronized in the I/O block, the signals can be used internally without further consideration of their clock relative timing, except as it applies to the internal logic and routing path delays.

Device Performance

The single parameter which most accurately describes the overall performance of the Logic Cell Array is the maximum toggle rate for a logic block storage element configured as a toggle flip-flop. The configuration for determining the toggle performance of the Logic Cell Array is shown in Figure 19. The clock for the storage element is provided by the global clock buffer and the flip-flop output Q is fed back through the combinatorial logic to form the data input for the next clock edge. Using this arrangement, flip-flops in the Logic Cell Array can be toggled at clock rates from 33-70 MHz, depending on the speed grade used.

Actual Logic Cell Array performance is determined by the critical path speed, including both the speed of the logic and storage elements in that path, and the speed of the particular network routing. Figure 20 shows a typical system logic configuration of two flip-flops with an extra combinatorial level between them. Depending on speed grade, system clock rates to 35 MHz are practical for this logic. To allow the user to make the best use of the capabilities of the device, the delay calculator in the XACT Development System determines worst-case path delays using actual impedance and loading information.

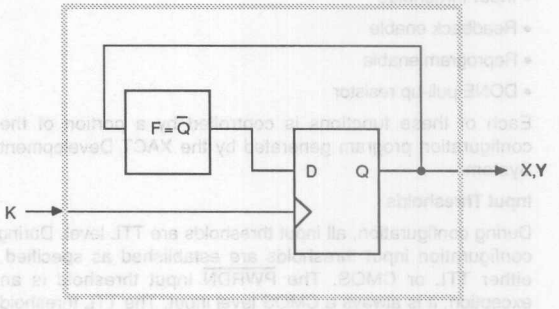


Figure 19. Logic Block Configuration for Toggle Rate Measurement

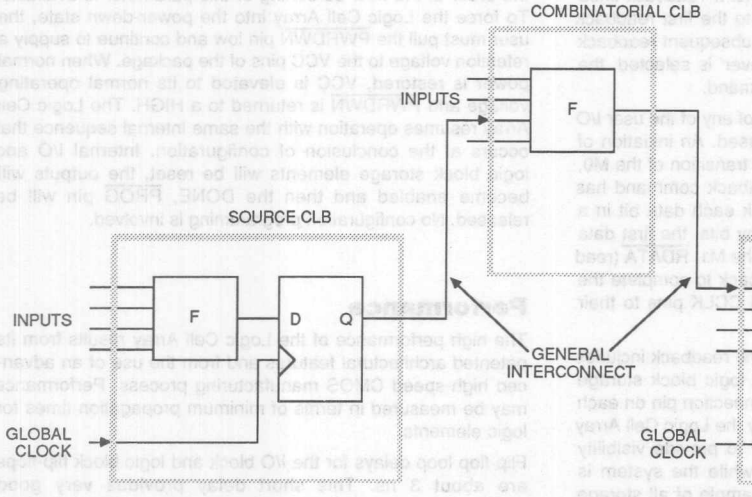
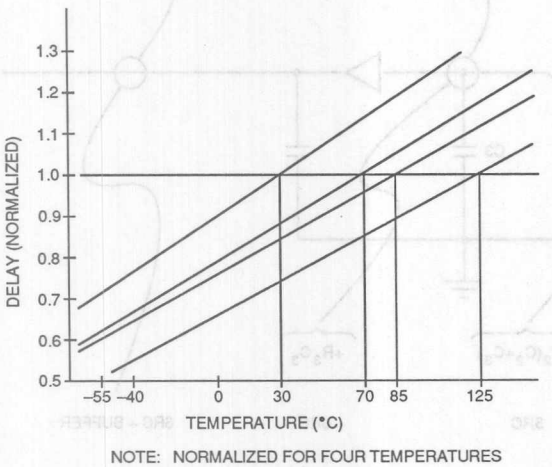


Figure 20. Typical Logic Path

Logic Block Performance

Logic Block propagation times are measured from the interconnect point at the input of the combinatorial logic to the output of the block in the interconnect area. Combinatorial performance is independent of logic function because of the table look-up based implementation. Timing is different when the combinatorial logic is used in conjunction with the storage element. For the combinatorial logic function driving the data input of the storage element, the critical timing is data set-up relative to the clock edge provided to the storage element. The delay from the clock source to the output of the logic block is critical in the timing of signals produced by storage elements. The loading on a logic block output is limited only by the additional propagation delay of the interconnect network. Performance of the logic block is a function of supply voltage and temperature, as shown in Figures 21 and 22.



NOTE: NORMALIZED FOR FOUR TEMPERATURES

Figure 21. Delay vs. Temperature

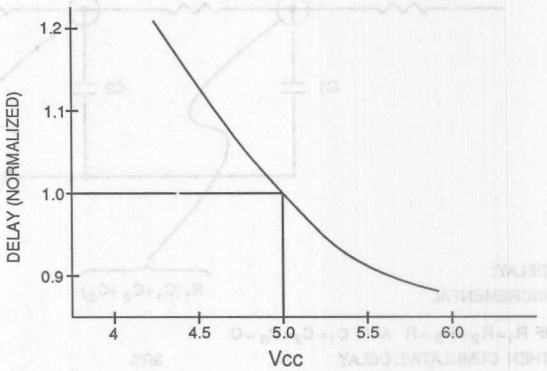


Figure 22. Delay vs. Power Supply

Interconnect Performance

Interconnect performance depends on the routing resource used to implement the signal path. As discussed earlier, direct interconnect from block to block provides a minimum delay path for a signal.

The single metal segment used for long lines exhibits low resistance from end to end, but relatively high capacitance. Signals driven through a programmable switch will have the additional impedance of the switch added to their normal drive impedance.

General-purpose interconnect performance depends on the number of switches and segments used, the presence of the bidirectional repowering buffers and the overall loading on the signal path at all points along the path. In calculating the worst-case delay for a general interconnect path, the delay calculator portion of the XACT development system accounts for all of these elements. As an approximation, interconnect delay is proportional to the summation of totals of local metal segments beyond each programmable switch. In effect, the delay is a sum of R-C delays each approximated by an R times the total C it drives. The R of the switch and the C of the interconnect are functions of the particular device performance grade. For a string of three local interconnects, the approximate delay at the first segment, after the first switch resistance, would be three units; an additonal two delay units after the next switch plus an additional delay after the last switch in the chain. The interconnect R-C chain terminates at each repowering buffer. Nearly all of the capacitance is in the interconnect metal and switches; the capacitance of the block inputs is not significant. Figure 23 shows an estimation of this delay.

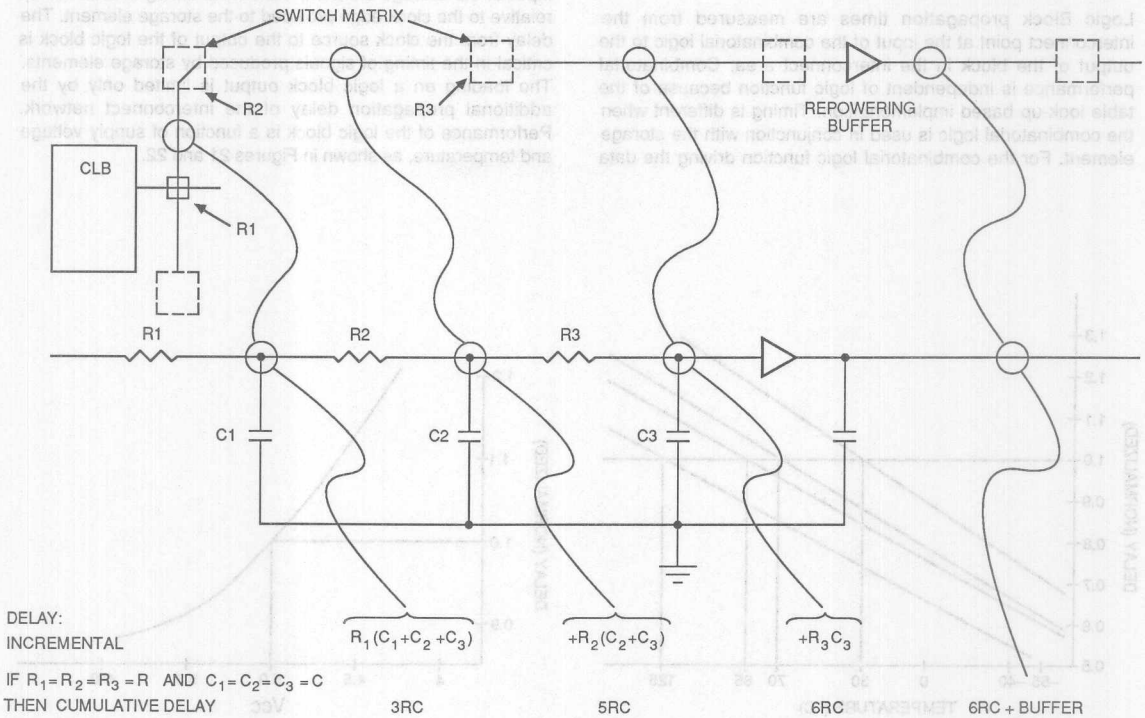


Figure 23. Interconnection Delay Example

Development System

To support designers using the Logic Cell Array , Monolithic Memories provides a basic development system with several options for additional productivity. The XACT system provides the following:

- Graphic-driven design entry
- Schematic entry
- Interactive timing delay calculations
- Macrocell library support, both for standard Monolithic Memories supplied functions and user-defined functions
- Design entry checking for consistency and completeness
- Automatic design documentation generation
- Automatic placement and routing
- Simulation interface support, including automatic netlist (circuit description) and timing extraction
- In-circuit emulation for multiple devices

The host system on which the XACT system operates is an IBM™ PC-XT™ or PC-AT™ or compatible system with MS-DOS™ 2.1 or higher. Color graphics is required as well as 640 K bytes of internal RAM (an Expanded Memory Specification (EMS) card with 256 K bytes of memory is required for the M2018). A complete system requires one parallel I/O port and two serial ports and a mouse.

For more detailed information of the XACT Development System, please refer to Logic Cell Array Development System Datasheet.

48-PIN DIP	68-PIN PLCC	68-PIN PGA	CONFIGURATION MODE: ≪M2: M1: M0≫				USER OPERATION
			SLAVE <1:1:1>	PERIPHERAL <1:0:1>	MASTER-HIGH <1:1:0>	MASTER-LOW <1:0:0>	
	1	B6	GND				
	2	A6	≪HIGH≫		A13 (O)		I/O
1	3	B5			A6 (O)		
	4	A5			A12 (O)		
2	5	B4			A7 (O)		
3	6	A4			A11 (O)		
4	7	B3			A8 (O)		
5	8	A3			A10 (O)		
6	9	A2			A9 (O)		
7	10	B2	$\overline{\text{PWRDWN}}$ (I)				
8	11	B1	≪HIGH≫				I/O
	12	C2					
9	13	C1					
	14	D2					
10	15	D1					
	16	E2					
11	17	E1					
12	18	F2	VCC				
13	19	F1	≪HIGH≫				I/O
	20	G2					
14	21	G1					
	22	H2					
15	23	H1					
16	24	J2					
17	25	J1	M1 (HIGH)	M1 (LOW)	M1 (HIGH)	M1 (LOW)	$\overline{\text{RDATA}}$ (O)
18	26	K1	M0 (HIGH)	M0 (HIGH)	M0 (LOW)	M0 (LOW)	RTRIG (I)
19	27	K2	M2 (HIGH)				I/O
20	28	L2	HDC (HIGH)				
	29	K3	≪HIGH≫				
21	30	L3	LDC (LOW)				
	31	K4	≪HIGH≫				
22	32	L4					
	33	K5					
23	34	L5					

≪HIGH≫ is high impedance with a 20 to 50-KΩ internal pull-up resistor during configuration

Table 2a. M2064 Pin Assignments

(continued on next page)

48-PIN DIP	68-PIN PLCC	68-PIN PGA	CONFIGURATION MODE: ≪M2: M1: M0≫				USER OPERATION
			SLAVE <1:1:1>	PERIPHERAL <1:0:1>	MASTER-HIGH <1:1:0>	MASTER-LOW <1:0:0>	
24	35	K6	GND				
	36	L6	≪HIGH≫				
25	37	K7					
	38	L7					
26	39	K8					I/O
27	40	L8					
28	41	K9			D7 (I)		
29	42	L9			D6 (I)		
30	43	L10					XTL2 or I/O
31	44	K10	$\overline{\text{RESET}}$ (I)				
32	45	K11	DONE (O)				$\overline{\text{PROG}}$ (I)
33	46	J10	≪HIGH≫				XTL1 or I/O
	47	J11					
34	48	H10			D5 (I)		
	49	H11					I/O
35	50	G10		$\overline{\text{CS0}}$ (I)	D4 (I)		
36	51	G11		$\overline{\text{CS1}}$ (I)	D3 (I)		
	52	F10	VCC				
	53	F11	≪HIGH≫				
37	54	E10		CS2 (I)	D2 (I)		
	55	E11					
38	56	D10		$\overline{\text{WRT}}$ (I)	D1 (I)		I/O
39	57	D11			$\overline{\text{RCLK}}$		
40	58	C10	DIN (I)		D0 (I)		
41	59	C11	DOUT (O)				
42	60	B11	CCLK (I)	CCLK (O)			CCLK (I)
43	61	B10	≪HIGH≫		A0 (O)		
44	62	A10			A1 (O)		
45	63	B9			A2 (O)		
46	64	A9			A3 (O)		I/O
	65	B8			A15 (O)		
47	66	A8			A4 (O)		
	67	B7			A14 (O)		
48	68	A7			A5 (O)		

≪HIGH≫ is high impedance with a 20 to 50-KΩ internal pull-up resistor during configuration

Table 2a. M2064 Pin Assignments (continued)

68-PIN PLCC	68-PIN PGA	84-PIN PLCC	84-PIN PGA	CONFIGURATION MODE: ≪M2: M1: M0≫ SLAVE <1:1:1>	PERIPHERAL <1:0:1>	MASTER-HIGH <1:1:0>	MASTER-LOW <1:0:0>	USER OPERATION
1	B6	1	C6	GND				
2	A6	2	A6	≪HIGH≫		A13 (O)		I/O
		3	A5					
		4	B5					
3	B5	5	C5			A6 (O)		
4	A5	6	A4			A12 (O)		
5	B4	7	B4			A7 (O)		
6	A4	8	A3			A11 (O)		
7	B3	9	A2			A8 (O)		
8	A3	10	B3			A10 (O)		
9	A2	11	A1			A9 (O)		
10	B2	12	B2	$\overline{\text{PWRDWN}}$ (I)				
11	B1	13	C2	≪HIGH≫				I/O
12	C2	14	B1					
13	C1	15	C1					
14	D2	16	D2					
15	D1	17	D1					
		18	E3					
16	E2	19	E2					
		20	E1					
17	E1	21	F2					
18	F2	22	F3	VCC				
19	F1	23	G3	≪HIGH≫				I/O
		24	G1					
20	G2	25	G2					
		26	F1					
21	G1	27	H1					
22	H2	28	H2					
23	H1	29	J1					
24	J2	30	K1					
25	J1	31	J2	M1 (HIGH)	M1 (LOW)	M1 (HIGH)	M1 (LOW)	$\overline{\text{RDATA}}$ (O)
26	K1	32	L1	M0 (HIGH)	M0 (HIGH)	M0 (LOW)	M0 (LOW)	RTRIG (I)
27	K2	33	K2	M2 (HIGH)				
28	L2	34	K3	HDC (HIGH)				
29	K3	35	L2	≪HIGH≫				
30	L3	36	L3	LDC (LOW)				
31	K4	37	K4	≪HIGH≫				I/O
32	L4	38	L4					
		39	J5					
33	K5	40	K5					
34	L5	41	L5					
		42	K6					

≪HIGH≫ is high impedance with a 20 to 50-KΩ internal pull-up resistor during configuration

Table 2b. M2018 Pin Assignments (continued on next page)

68-PIN PLCC	68-PIN PGA	84-PIN PLCC	84-PIN PGA	CONFIGURATION MODE: ≪M2: M1: M0≫ SLAVE <1:1:1>	PERIPHERAL <1:0:1>	MASTER-HIGH <1:1:0>	MASTER-LOW <1:0:0>	USER OPERATION
35	K6	43	J6	GND				
		44	J7					
36	L6	45	L7					
37	K7	46	K7					
38	L7	47	L6	≪HIGH≫				
		48	L8					I/O
39	K8	49	K8					
40	L8	50	L9					
41	K9	51	L10			D7 (I)		
42	L9	52	K9			D6 (I)		
43	L10	53	L11					XT2 or I/O
44	K10	54	K10	$\overline{\text{RESET}}$ (I)				
45	K11	55	J10	DONE (0)				$\overline{\text{PROG}}$ (1)
46	J10	56	K11					XTL1 or I/O
47	J11	57	J11					
48	H10	58	H10	≪HIGH≫		D5 (I)		
		59	H11					
49	H11	60	F10					I/O
		61	G10					
50	G10	62	G11		$\overline{\text{CS0}}$ (I)	D4 (I)		
51	G11	63	G9		$\overline{\text{CS1}}$ (I)	D3 (I)		
52	F10	64	F9	VCC				
53	F11	65	F11					
54	E10	66	E11		CS2 (I)	D2 (I)		
		67	E10					
55	E11	68	E9	≪HIGH≫				
		69	D11					I/O
56	D10	70	D10		$\overline{\text{WRT}}$ (I)	D1 (I)		
57	D11	71	C11			$\overline{\text{RCLK}}$		
58	C10	72	B11	DIN (I)		D0 (I)		
59	C11	73	C10	DOUT (O)				
60	B11	74	A11	CCLK (I)	CCLK (O)			CCLK (I)
61	B10	75	B10			A0 (O)		
62	A10	76	B9			A1 (O)		
63	B9	77	A10			A2 (O)		
64	A9	78	A9			A3 (O)		
65	B8	79	B8	≪HIGH≫		A15 (O)		I/O
66	A8	80	A8			A4 (O)		
67	B7	81	B6			A14 (O)		
		82	B7					
		83	A7					
68	A7	84	C7			A5 (O)		

≪HIGH≫ is high impedance with a 20 to 50-KΩ internal pull-up resistor during configuration

Table 2b. M2018 Pin Assignments (continued)

Absolute Maximum Ratings*

Supply voltage V_{CC} ..... −0.5 V to 7 V
Power down V_{CC} ..... 2 V to 7 V
Input voltage ..... −0.5 V to V_{CC} 0.5 V
Voltage applied to three-state output ..... −0.5 V to V_{CC} 0.5 V
Storage temperature range ..... −65° C to +150° C
Lead temperature (soldering, 10 seconds) ..... 260° C

* Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only and functional operation of the device at these or any other conditions beyond those listed under "Recommended Operating Conditions" is not implied. Exposure to "Absolute Maximum Ratings" conditions for extended periods of time may affect device reliability.

Operating Conditions

SYMBOL	PARAMETER	MIN	TYP	MAX	UNIT
V_{CC}	Supply voltage relative to GND	4.75		5.25	V
V_{IHT}	High level input voltage—TTL configuration	2.0		V_{CC}	V
V_{IHC}	High level input voltage—CMOS configuration	0.7 V_{CC}		V_{CC}	V
V_{ILT}	Low level input voltage—TTL configuration	0		0.8	V
V_{ILC}	Low level input voltage—CMOS configuration	0		0.2 V_{CC}	V
I_{IT}	Input leakage current—TTL configuration	±10			μA
I_{IC}	Input leakage current—CMOS configuration	±10			μA
I_{OZ}	Three-state output off current (V_{CC} = 5.5 V)	±10			μA
t_{OP}	Operating free-air temperature	0		70	°C

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER		TEST CONDITION		MIN	TYP	MAX	UNIT
V_{OH}	High level output voltage		V_{CC} = 4.75 V	I_{OH} = −4.0 mA	3.86			V
V_{OL}	Low level output voltage		V_{CC} = 4.75 V	I_{OL} = 4.0 mA			0.32	V
I_{CCO}	Quiescent operating power supply current	CMOS inputs	V_{CC} = 5.0 V				5	mA
		TTL inputs	V_{CC} = 5.0 V				10	mA
I_{CCPD}	Power down supply current		V_{CC} = 5.0 V			0.5		mA

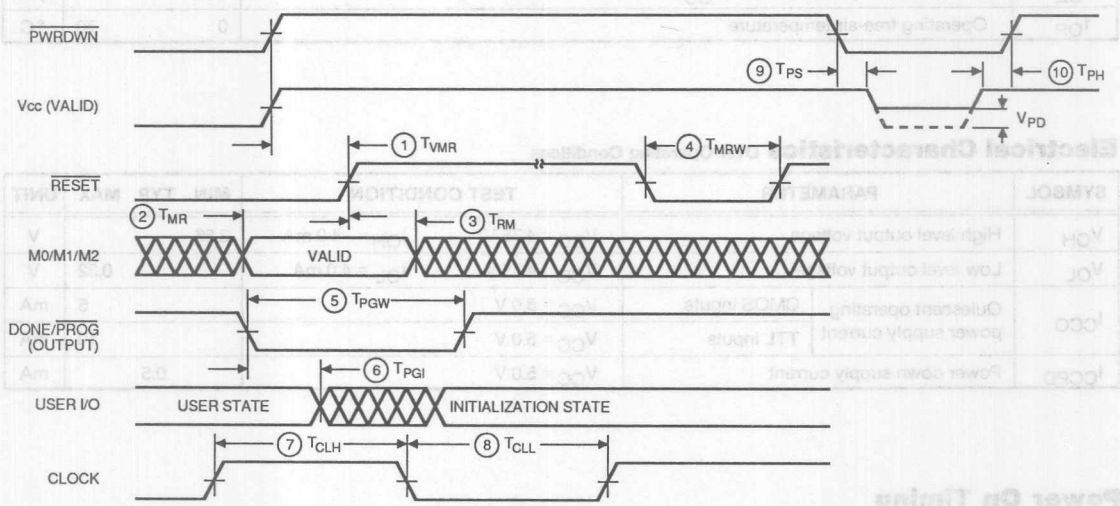
Power On Timing

The LCAs contain on-chip reset timing logic for power-up operation. To insure proper master mode system operation, VCC must rise from 2.0 V to minimum specification level in 10 ms or less. For other modes, initiation of configuration must be delayed for 60 ms after VCC reaches the minimum specified level.

Switching Characteristics — General

SYMBOL	DESCRIPTION		-33		-50		-70		UNIT
			MIN	MAX	MIN	MAX	MIN	MAX	
t_{VMR} ①	$\overline{RESET}$ (2)	V_{CC} setup (2.0 V)	150		150		150		ns
t_{MR} ②		M2, M1, M0 setup	60		60		60		ns
t_{RM} ③		M2, M1, M0 hold	60		60		60		ns
t_{MRW} ④		Width (LOW)	150		150		150		ns
t_{PGW} ⑤	DONE/$\overline{PROG}$	Program width (LOW)	6		6		6		μs
t_{PGI} ⑥		Initialization		7		7		7	μs
t_{CLH} ⑦	CLOCK	Clock (HIGH)	12		8		7		ns
t_{CLL} ⑧		Clock (LOW)	12		8		7		ns
t_{PS} ⑨	$\overline{PWR\ DWN}$	Setup to V_{CC}	0		0		0		ns
t_{PH} ⑩		Hold from V_{CC}	0		0		0		ns
V_{PD}		Power Down	2.0		2.0		2.0		V

Notes: 1. V_{CC} must rise from 2.0 Volts to V_{CC} minimum in lest than 10 ms for master mode.
2. $\overline{RESET}$ timing relative to power-on and valid mode lines (M0, M1, M2) is relevant only when $\overline{RESET}$ is used to delay configuration.
3. Minimum CLOCK widths for the auxilliary buffer are 1.25 times the t_{CLH}, t_{CLL}.



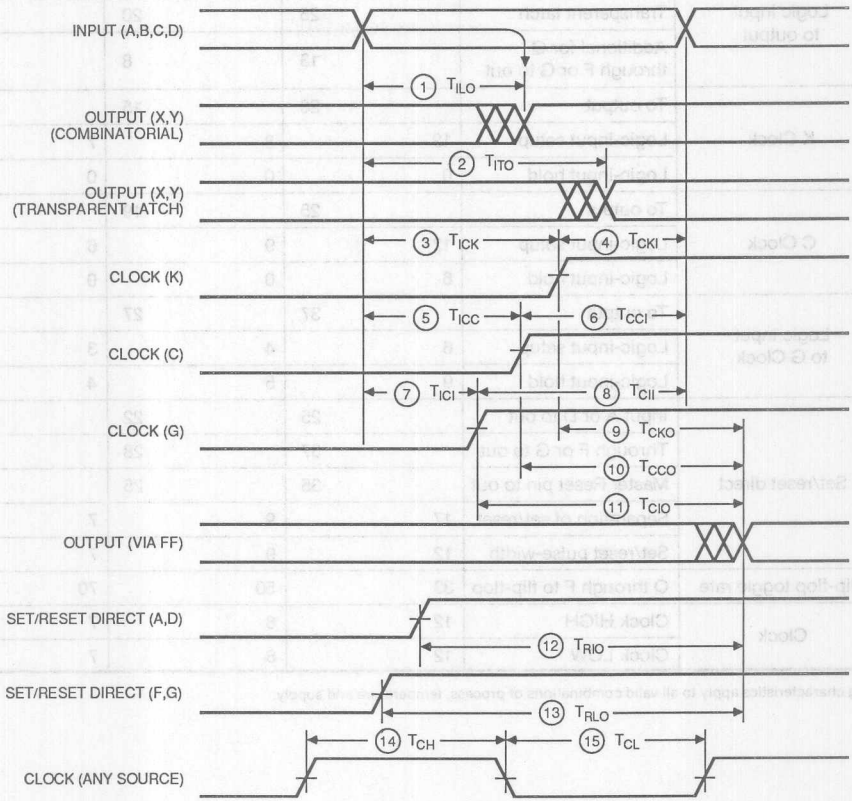
Switching Characteristics — CLB

SYMBOL	DESCRIPTION		-33		-50		-70		UNIT
			MIN	MAX	MIN	MAX	MIN	MAX	
t_{ILO} ①	Logic input to output	Combinatorial		20		15		10	ns
t_{ITO} ②		Transparent latch		25		20		14	ns
t_{QLO}		Additional for Q through F or G to out		13		8		6	ns
t_{CKO} ⑨	K Clock	To output		20		15		10	ns
t_{ICK} ③		Logic-input setup	12		8		7		ns
t_{CKI} ④		Logic-input hold	0		0		0		ns
t_{CCO} ⑩	C Clock	To output		25		19		13	ns
t_{ICC} ⑤		Logic-input setup	12		9		6		ns
t_{CCI} ⑥		Logic-input hold	6		0		0		ns
t_{CIO} ⑪	Logic input to G Clock	To output		37		27		20	ns
t_{ICI} ⑦		Logic-input setup	6		4		3		ns
t_{CII} ⑧		Logic-input hold	9		5		4		ns
t_{RIO} ⑫	Set/reset direct	Input A or D to out		25		22		16	ns
t_{RLO} ⑬		Through F or G to out		37		28		21	ns
t_{MRQ}		Master Reset pin to out		35		25		20	ns
t_{RS}		Separation of set/reset	17		9		7		ns
t_{RPW}		Set/reset pulse-width	12		9		7		ns
F_{CLK}	Flip-flop toggle rate	Q through F to flip-flop	33		50		70		MHz
t_{CH} ⑭	Clock	Clock HIGH	12		8		7		ns
t_{CL} ⑮		Clock LOW	12		8		7		ns

Note: All switching characteristics apply to all valid combinations of process, temperature and supply.

Cross Reference Guide

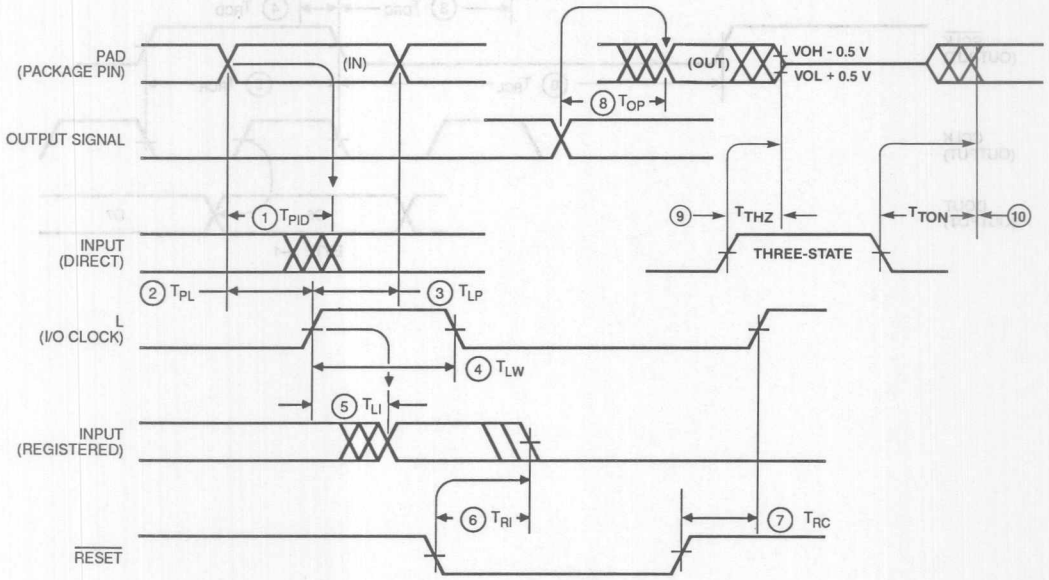
XILINX	MMI	V_{CC}		F_{MAX}
		MIN	MAX	MIN
XC2064-1		4.5 V	5.5 V	20 MHz
	M2064-20	4.75 V	5.25 V	20 MHz
XC2064-2		4.5 V	5.5 V	33 MHz
XC2064-33	M2064-33	4.75 V	5.25 V	33 MHz
XC2064-50	M2064-50	4.75 V	5.25 V	50 MHz
XC2064-70	M2064-70	4.75 V	5.25 V	70 MHz
XC2018-33	M2018-33	4.75 V	5.25 V	33 MHz
XC2018-50	M2018-50	4.75 V	5.25 V	50 MHz
XC2018-70	M2018-70	4.75 V	5.25 V	70 MHz


INPUT (A,B,C,D)
1 TILO
OUTPUT (X,Y)
(COMBINATORIAL)
2 TITO
OUTPUT (X,Y)
(TRANSPARENT LATCH)
3 TICK
4 TCKI
CLOCK (K)
5 TICC
6 TCCI
CLOCK (C)
7 TICI
8 TCII
CLOCK (G)
9 TCKO
10 TCCO
11 TCIO
OUTPUT (VIA FF)
SET/RESET DIRECT (A,D)
12 TRIO
SET/RESET DIRECT (F,G)
13 TRLO
14 TCH
15 TCL
CLOCK (ANY SOURCE)

Switching Characteristics — IOB

SYMBOL	DESCRIPTION	-33		-50		-70		UNIT
		MIN	MAX	MIN	MAX	MIN	MAX	
t_{PID} ①	Pad (package pin) to input (direct)		12		8		6	ns
t_{LI} ⑤	I/O Clock to input (storage)		20		15		11	ns
t_{PL} ②	I/O Clock to pad-input setup	12		8		6		ns
t_{LP} ③	I/O Clock to pad-input hold	0		0		0		ns
t_{LW} ④	I/O Clock pulse width	12		9		7		ns
	I/O Clock frequency	33		50		70		MHz
t_{OP} ⑧	Output to pad (output enabled)		15		12		9	ns
t_{THZ} ⑨	Three-state to pad begin hi-Z		25		20		15	ns
t_{TON} ⑩	Three-state to pad end hi-Z		25		20		15	ns
t_{RI} ⑥	$\overline{\text{RESET}}$ to input (storage)		40		30		25	ns
t_{RC} ⑦	$\overline{\text{RESET}}$ to input clock		35		25		20	ns

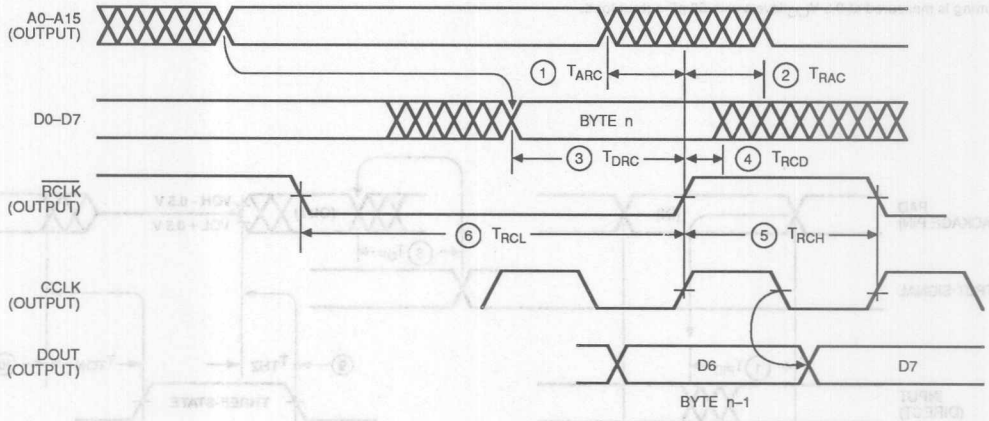
Note: Timing is measured at 0.5 V_{CC} levels with 50 pF output load.



Switching Characteristics — Programming — Master Mode

SYMBOL	DESCRIPTION		-33		-50		-70		UNIT
			MIN	MAX	MIN	MAX	MIN	MAX	
t_{ARC} ①	RCLK	From address invalid		0		0		0	ns
t_{RAC} ②		To address valid		200		200		200	ns
t_{DRC} ③		To data setup	60		60		60		ns
t_{RCD} ④		To data hold	0		0		0		ns
t_{RCH} ⑤		$\overline{RCLK}$ HIGH	600		600		600		ns
t_{RCL} ⑥		$\overline{RCLK}$ LOW	4.0		4.0		4.0		μs

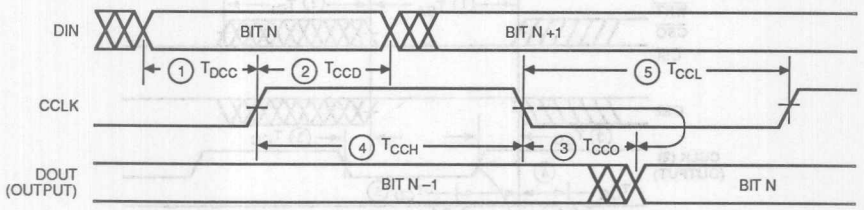
Notes: 1. CCLK and DOUT timing are the same as for slave mode.
2. At power up, V_{CC} must rise from 2.0 V to V_{CC} minimum in less than 10 ms.



Switching Characteristics — Programming — Slave Mode

SYMBOL	DESCRIPTION	-33		-50		-70		UNIT
		MIN	MAX	MIN	MAX	MIN	MAX	
t_{CCO} ③	CCLK to DOUT		65		65		65	ns
t_{DCC} ①	CCLK DIN setup	0		0		0		ns
t_{CCD} ②	CCLK DIN hold	40		40		40		ns
t_{CCH} ④	CCLK HIGH time	0.25		0.25		0.25		μs
t_{CCL} ⑤	CCLK LOW time	0.25	5.0	0.25	5.0	0.25	5.0	μs
F_{CC}	CCLK frequency		2		2		2	MHz

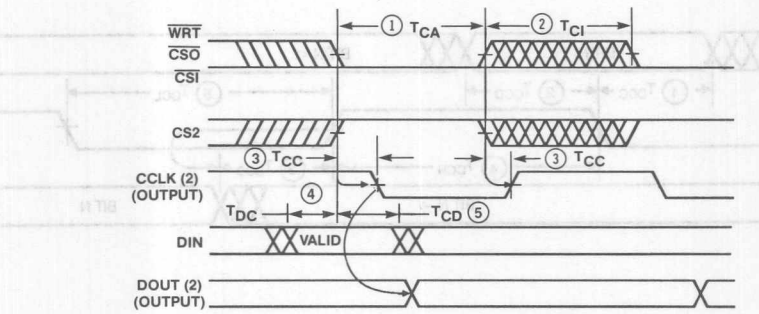
Note: Configuration must be delayed at least 40 ms after V_{CC} minimum.



Switching Characteristics — Programming — Peripheral Mode

SYMBOL		DESCRIPTION	-33		-50		-70		UNIT
			MIN	MAX	MIN	MAX	MIN	MAX	
t_{CA} ①	Controls[1] ($\overline{CS0}$, $\overline{CS1}$, CS2, $\overline{WRT}$)	Active (last active input to first inactive)	0.25	5.0	0.25	5.0	0.25	5.0	μs
t_{CI} ②		Inactive (first inactive input to last active)	0.25		0.25		0.25		μs
t_{CCC} ③		CCLK[2]		75		75		75	ns
t_{DC} ④		DIN setup	35		35		35		ns
t_{CD} ⑤		DIN hold	5		5		5		ns

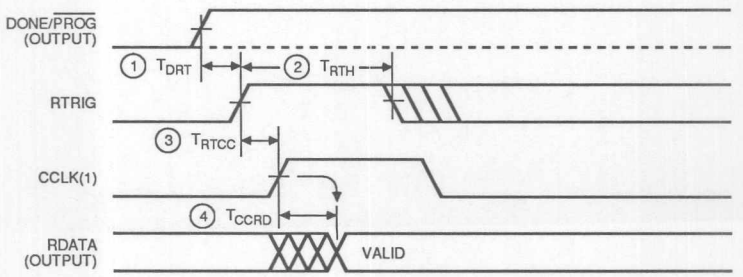
Notes: 1. Peripheral mode timing determined from last control signal of the logical AND of ($\overline{CS0}$, $\overline{CS1}$, CS2, $\overline{WRT}$) to transition to active or inactive state.
2. CCLK and DOUT timing are the same as for slave mode.
3. Configuration must be delayed at least 40 ms after V_{CC} minimum.



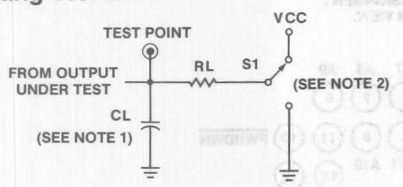
Switching Characteristics — Program Readback

SYMBOL	DESCRIPTION		-33		-50		-70		UNIT
			MIN	MAX	MIN	MAX	MIN	MAX	
t_{DRT} ①	RTRIG	PROG setup	300		300		300		ns
t_{RTH} ②		RTRIG HIGH	250		250		250		ns
t_{RTCC} ③	CCLK	RTRIG setup	100		100		100		ns
t_{CCRD} ④		RDATA delay		100		100		100	ns

Notes: 1. CCLK and DOUT timing are the same as for slave mode.
2. DONE/PROG output/input must be HIGH (device programmed) prior to a positive transition of RTRIG (M0).



Switching Test Load



Note: CL includes probe and jig capacitance.

CL = 50 pF
RL = 1K

Design Aids

Designing with the Logic Cell Array is similar to using conventional MSI elements or gate array macrocells. The first step is to partition the desired logic design into Logic Blocks and I/O blocks, usually based on shared input variables or efficient use of flip-flop and combinatorial logic. Following a plan for placement of the blocks, the design information may be entered using the interactive Graphic Design Editor. The design information includes both the functional specifications for each block and a definition of the interconnection networks. A macrocell library provides a simplified entry of commonly-used logic functions. As an alternative to interactive block placement and configuration, a schematic may be created using elements from the macrocell library. Automatic placement and routing is available for either method of design entry. After routing the interconnections, various checking stages and processing of that data are performed to ensure that the design is correct. Design changes may be implemented in minutes. The design file is used to generate the programming data which can be downloaded directly into an LCA in the target system and operated. The program information may be used to program PROM, EPROM or ROM devices, or stored in some other media as needed by the final system.

Design verification may be accomplished by using the XLINX XACTOR™ In-Circuit Emulation System directly in the target system and/or the P-Silos™ logic simulator.

2

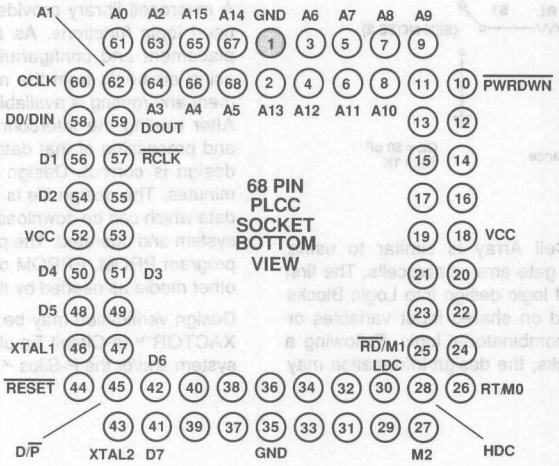
Recommended Sockets

The following sockets, with matching hole patterns, are available for PLCC devices.

	DESCRIPTION	VENDOR	PART NUMBER
68-pin	PCB solder tail, tin plate	AMP	821574-1
	Surface mount, tin plate	AMP	821542-1
	PCB solder tail, tin plate	Burndy*	QILE68P-410T
	PCB solder tail, tin plate	Midland-Ross*	709-2000-068-4-1-1
	PCB solder tail, tin plate	Methode*	213-068-001
	Surface mount, tin plate	Methode*	213-068-002
84-pin	PCB solder tail, tin plate	AMP	821573-1
	Surface mount, tin plate	AMP	821546-1
	PCB solder tail, tin plate	Burndy*	QILE84P-410T
	PCB solder tail, tin plate	Midland-Ross*	709-2000-084-4-1-1
	PCB solder tail, tin plate	Methode*	213-084-001
	Surface mount, tin plate	Methode*	213-084-002

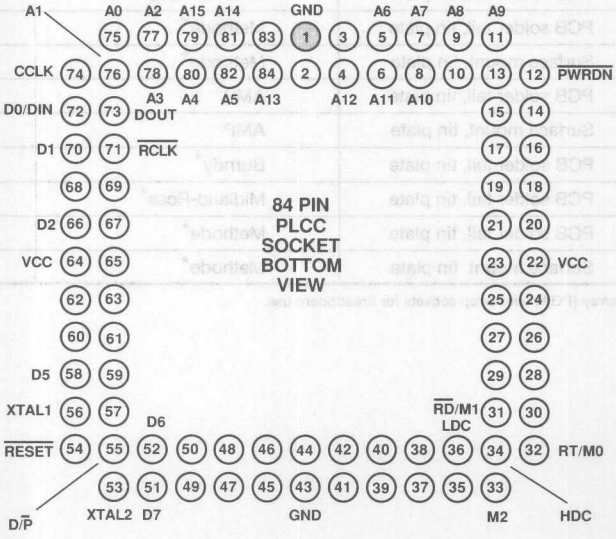
* Sockets will plug into pin-grid array (PGA) wire-wrap sockets for breadboard use.

M2064/18 PLCC SOCKET PIN ASSIGNMENT
WIRING REFERENCE. BOTTOM VIEW.



FOR EASE OF WIRING, AND PIN IDENTIFICATION, THE BOTTOM VIEW OF THE PLCC IS SHOWN ALONG WITH KEY PIN ASSIGNMENTS, SUCH AS ADDRESS, DATA, MODE, POWER AND CRYSTAL OSCILLATION INPUTS.

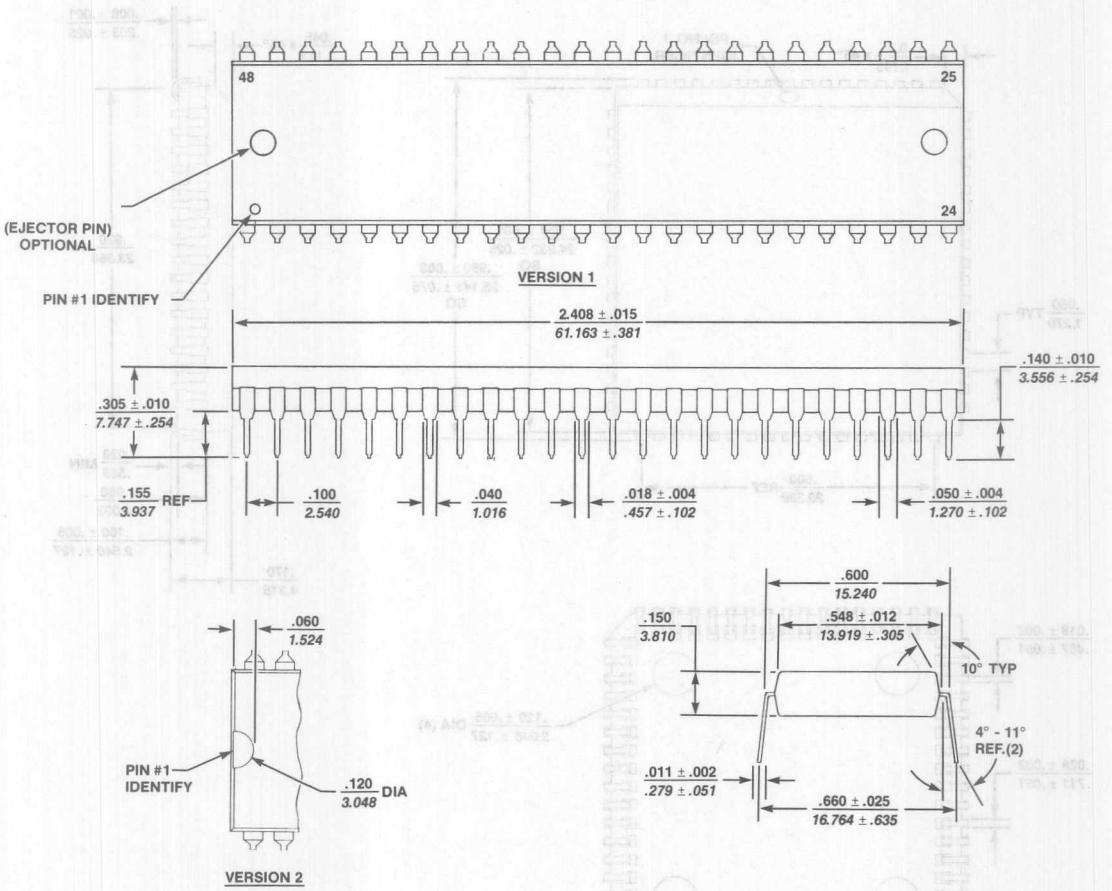
M2018 PLCC SOCKET PIN ASSIGNMENT
WIRING REFERENCE. BOTTOM VIEW.



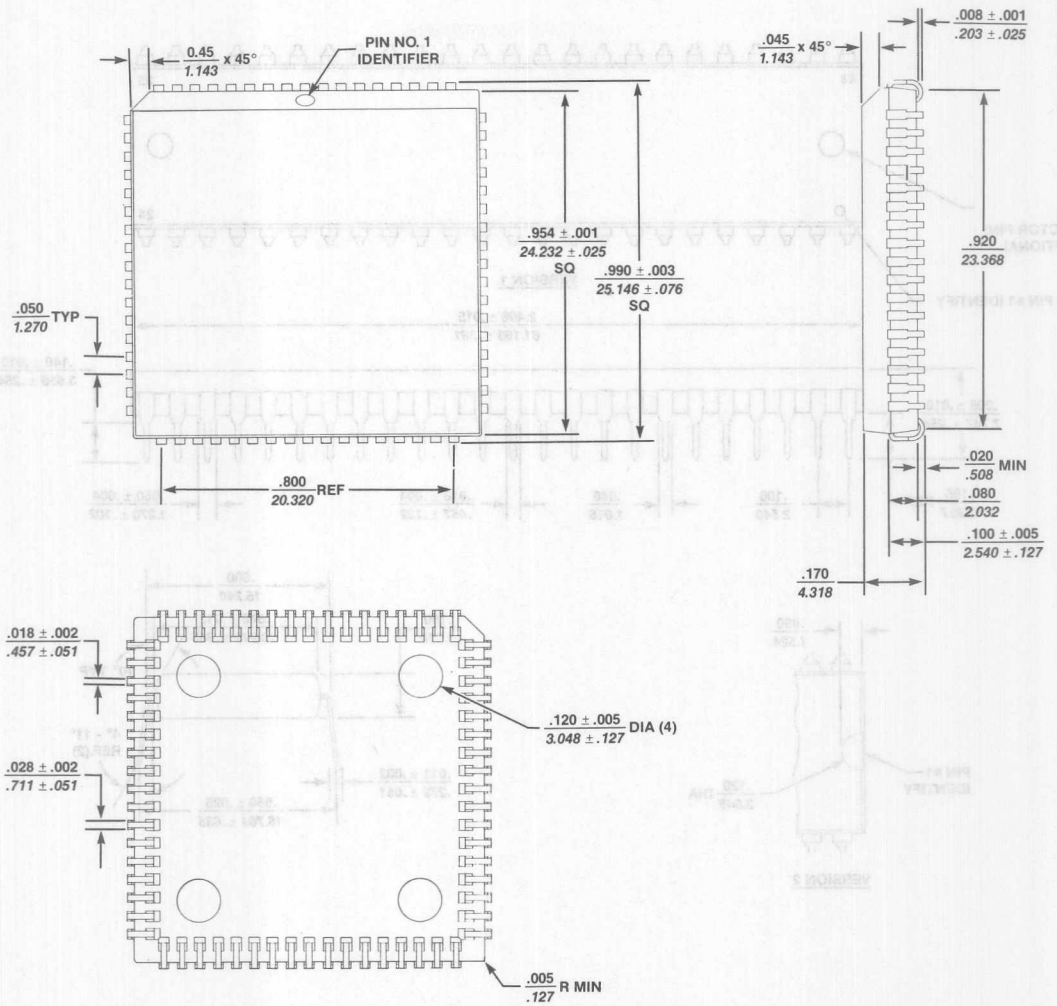
FOR EASE OF WIRING, AND PIN IDENTIFICATION, THE BOTTOM VIEW OF THE PLCC IS SHOWN ALONG WITH KEY PIN ASSIGNMENTS, SUCH AS ADDRESS, DATA, MODE, POWER AND CRYSTAL OSCILLATION INPUTS.

Package Drawing

48N Molded DIP
(9/16"x2-13/32")



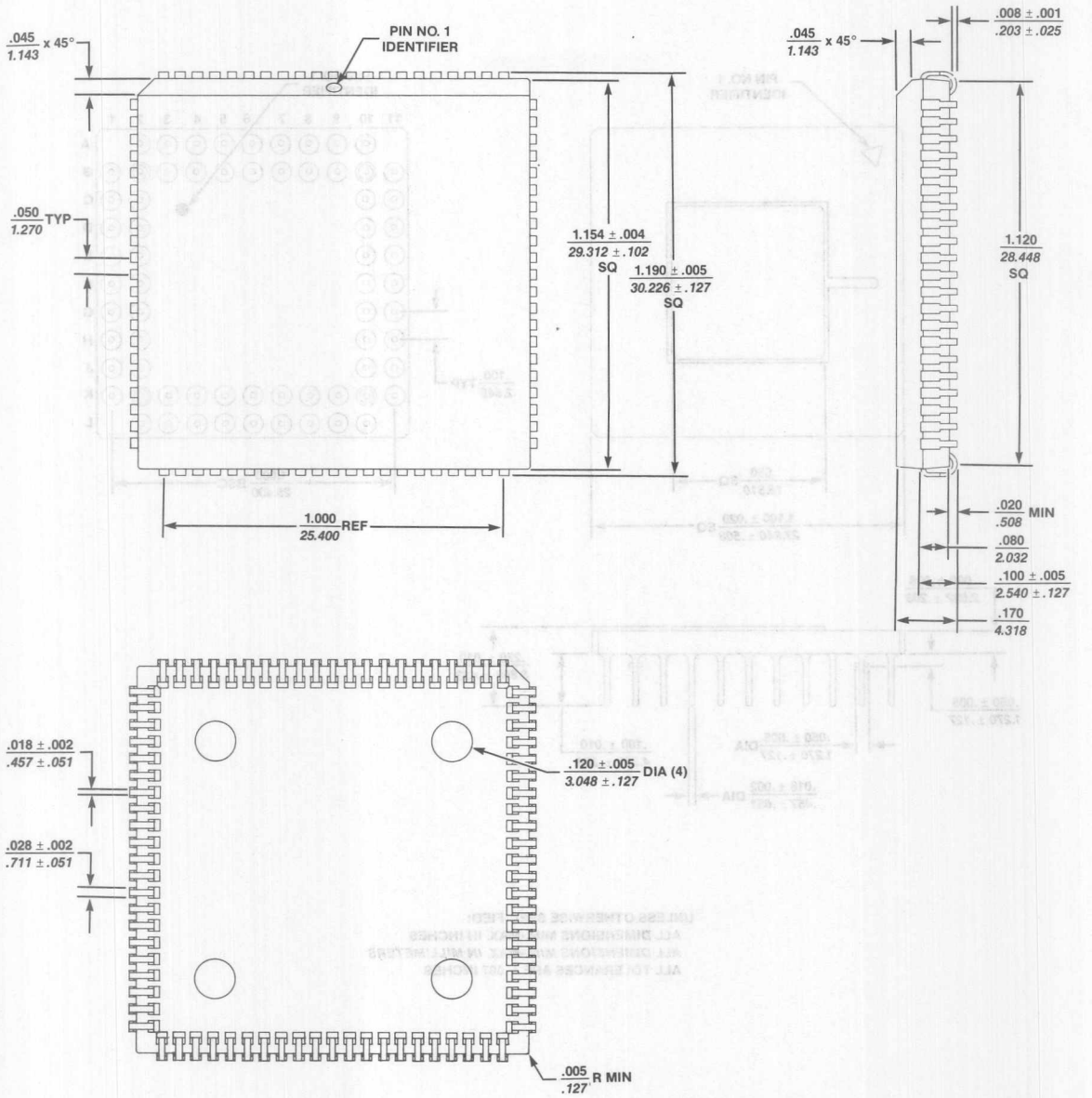
UNLESS OTHERWISE SPECIFIED:
ALL DIMENSIONS MIN.-MAX. IN INCHES
ALL DIMENSIONS MIN.-MAX. IN MILLIMETERS
ALL TOLERANCES ARE ± .007 INCHES



UNLESS OTHERWISE SPECIFIED:
ALL DIMENSIONS MIN.-MAX. IN INCHES
ALL DIMENSIONS MIN.-MAX. IN MILLIMETERS
ALL TOLERANCES ARE ± .007 INCHES

Package Drawing

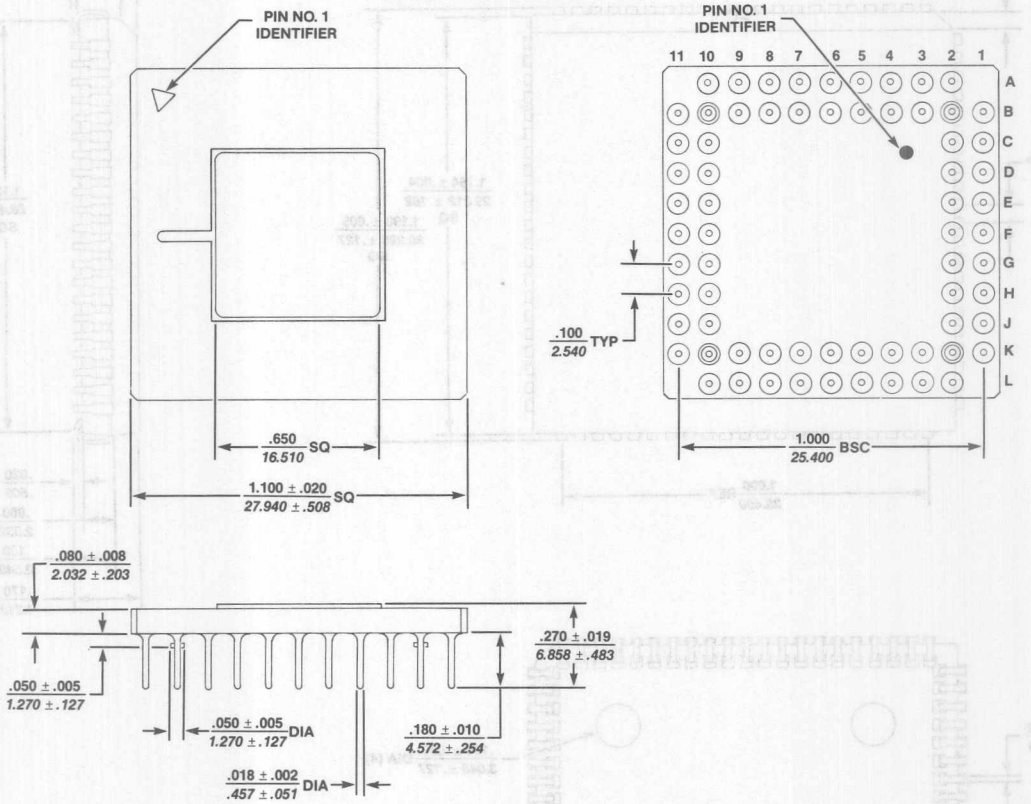
84NL Molded Chip Carrier



UNLESS OTHERWISE SPECIFIED:
ALL DIMENSIONS MIN.-MAX. IN INCHES
ALL DIMENSIONS MIN.-MAX. IN MILLIMETERS
ALL TOLERANCES ARE ± .007 INCHES

Package Drawing

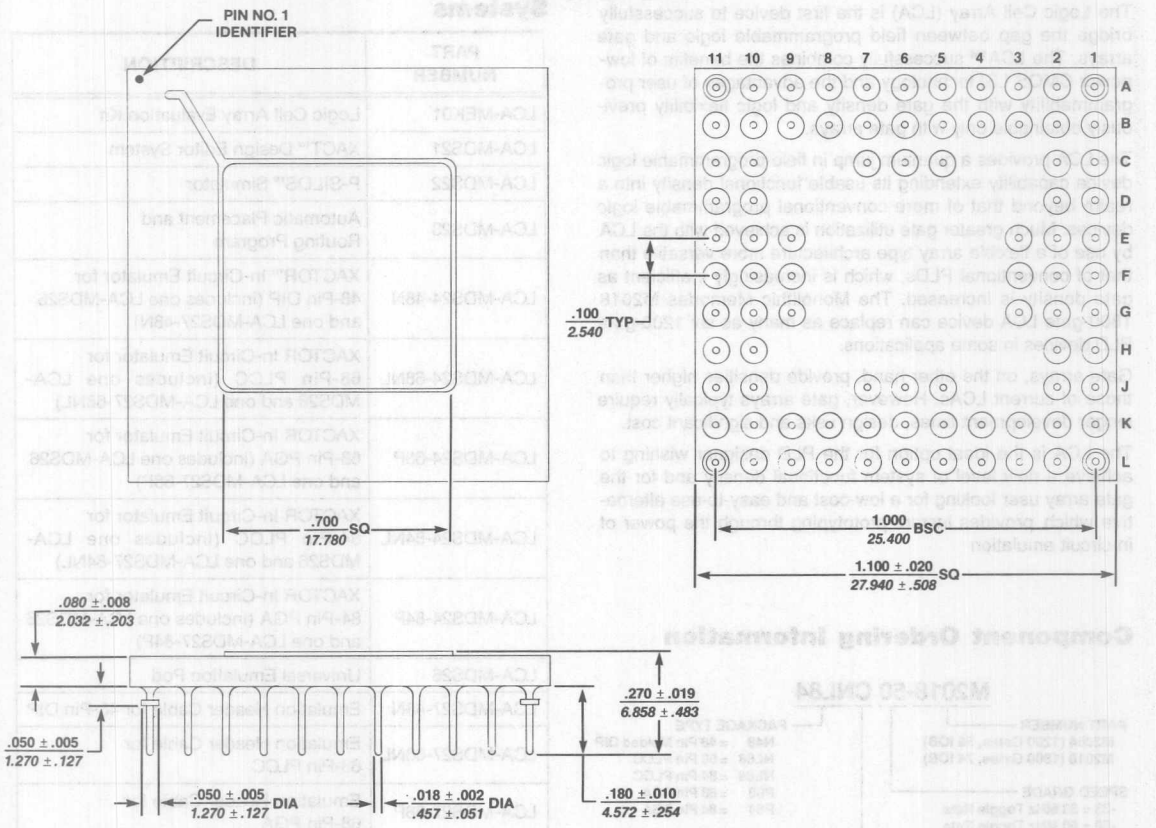
68P Ceramic Pin Grid Array



UNLESS OTHERWISE SPECIFIED:
ALL DIMENSIONS MIN.-MAX. IN INCHES
ALL DIMENSIONS MIN.-MAX. IN MILLIMETERS
ALL TOLERANCES ARE ± .007 INCHES

Package Drawing

84P Ceramic Pin Grid Array



UNLESS OTHERWISE SPECIFIED:
ALL DIMENSIONS MIN.-MAX. IN INCHES
ALL DIMENSIONS MIN.-MAX. IN MILLIMETERS
ALL TOLERANCES ARE ± .007 INCHES

2

Logic Cell™ Array and Development Systems

The Logic Cell Array

The Logic Cell Array (LCA) is the first device to successfully bridge the gap between field programmable logic and gate arrays. The LCA™ successfully combines the benefits of low-power CMOS LSI technology and the advantages of user programmability with the gate density and logic flexibility previously obtainable only with gate arrays.

The LCA provides a quantum jump in field-programmable logic device capability extending its usable functional density into a realm beyond that of more conventional programmable logic devices. Much greater gate utilization is achieved with the LCA by use of a flexible array type architecture more versatile than that of conventional PLDs, which is increasingly inefficient as gate density is increased. The Monolithic Memories M2018 1800-gate LCA device can replace as many as six 1200-gate PLD devices in some applications.

Gate arrays, on the other hand, provide densities higher than those of current LCAs. However, gate arrays typically require longer development times, design risks and significant cost.

The LCA is the ideal option for the PLD designer wishing to achieve a new level of system functional density and for the gate array user looking for a low-cost and easy-to-use alternative which provides instant prototyping through the power of in-circuit emulation

Component Ordering Information

M2018-50 CNL84

PART NUMBER
M2064 (1200 Gates, 58 IOB)
M2018 (1800 Gates, 74 IOB)

SPEED GRADE
-33 = 33 MHz Toggle Rate
-50 = 50 MHz Toggle Rate
-70 = 70 MHz Toggle Rate

PACKAGE TYPE
N48 = 48 Pin Molded DIP
NL68 = 68 Pin PLCC
NL84 = 84 Pin PLCC
P68 = 68 Pin PGA
P84 = 84 Pin PGA

TEMPERATURE RANGE
C = Commercial
M = Military

Package Availability

PART NUMBER	48-PIN PLASTIC DIP N48	68-PIN PLCC NL68	68-PIN PGA P68	84-PIN PLCC NL84	84-PIN PGA P84
M2064	X	X	X		
M2018		X	X	X	X

Ordering Information Development Systems

PART NUMBER	DESCRIPTION
LCA-MEK01	Logic Cell Array Evaluation Kit
LCA-MDS21	XACT™ Design Editor System
LCA-MDS22	P-SILOS™ Simulator
LCA-MDS23	Automatic Placement and Routing Program
LCA-MDS24-48N	XACTOR™ In-Circuit Emulator for 48-Pin DIP (includes one LCA-MDS26 and one LCA-MDS27-48N)
LCA-MDS24-68NL	XACTOR In-Circuit Emulator for 68-Pin PLCC (includes one LCA-MDS26 and one LCA-MDS27-68NL)
LCA-MDS24-68P	XACTOR In-Circuit Emulator for 68-Pin PGA (includes one LCA-MDS26 and one LCA-MDS27-68P)
LCA-MDS24-84NL	XACTOR In-Circuit Emulator for 84-Pin PLCC (includes one LCA-MDS26 and one LCA-MDS27-84NL)
LCA-MDS24-84P	XACTOR In-Circuit Emulator for 84-Pin PGA (includes one LCA-MDS26 and one LCA-MDS27-84P)
LCA-MDS26	Universal Emulation Pod
LCA-MDS27-48N	Emulation Header Cable for 48-Pin DIP
LCA-MDS27-68NL	Emulation Header Cable for 68-Pin PLCC
LCA-MDS27-68P	Emulation Header Cable for 68-Pin PGA
LCA-MDS27-84NL	Emulation Header Cable for 84-Pin PLCC
LCA-MDS27-84P	Emulation Header Cable for 84-Pin PGA
LCA-MDS31	FutureNet® DASH™ Schematic Design Entry Interface

Service Contracts

PART NUMBER	DESCRIPTION
LCA-MSC21	XACT Design Editor System (LCA-MDS21) Annual Support Agreement

TWX: 910-338-2376
2175 Mission College Blvd. Santa Clara, CA 95054-1592 Tel: (408) 970-9700 TWX: 910-338-2374
Monolithic Memories

Logic Cell Array M2064, M2018

Features

- Fully Programmable
  - I/O functions
  - Digital logic functions
  - Interconnections
- General purpose array architecture
- Complete user control of design cycle
- Compatible arrays with logic cell complexity equivalent to 1200 and 1800 usable gates
- Standard product availability
- 100% factory-tested
- Selectable configuration modes
- Low-power, CMOS, static memory technology
- Three performance options: 33, 50, and 70 MHz
- TTL or CMOS input threshold levels
- Complete development system support
  - XACT Design Editor
  - Macro Library
  - Timing analyzer
  - Design rules checker
  - Configuration file generator
  - Configuration file formatter
- Optional features
  - Schematic capture entry
  - XACTOR in-circuit emulator
  - Logic and timing simulator
  - Auto Place/Route

Description

The Logic Cell Array (LCA) is a high-density CMOS user-programmable logic device. The array architecture of the LCA allows the designer total flexibility and yields extremely high gate utilization. The LCA is composed of three configurable logic elements: Input/Output Blocks (IOBs), Configurable Logic Blocks (CLBs), and Programmable Interconnect. The XACT development system Design Editor provides a graphical interface to configure individual IOBs for external interface, define CLBs to implement internal logic, and assemble an internal network of interconnect to accomplish larger logic functions. The XACT Design Editor provides an interactive graphic design capture system with an automatic routing feature. Both logic simulation and emulation are available for design verification.

Programming

The Logic Cell Array's logic functions and interconnections are determined by a configuration program stored in internal static memory cells. On-chip logic provides for automatic loading of configuration data at power-up or on command. The program data can reside in an EEPROM, EPROM, or ROM on the circuit board or on a floppy disk or hard disk.

Several methods of automatically loading the required data are designed into the Logic Cell Array and are determined by logic levels applied to mode selection pins at configuration time. The form of the data may be either serial or parallel, depending on the configuration mode. The programming data are independent of the configuration mode selected.

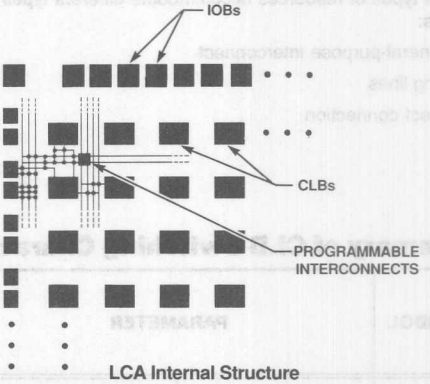
The Logic Cell Array is available in a variety of logic capacities, package styles, temperature ranges and speed grades.

Input/Output Block

Each user-configurable I/O block (IOB) provides an interface between the external package pin of the device and the internal logic. Each I/O block includes programmable input path and a programmable output buffer as shown in Figure 1. It also provides input clamping diodes to provide protection from electrostatic damage, and circuits to protect the LCA from latch-up due to input currents.

The input buffer portion of each I/O block provides threshold detection to translate external signals applied to the package pin to internal logic levels. The input buffer threshold of the I/O blocks can be programmed to be compatible with either TTL (1.4 V) or CMOS (2.2 V) levels.

Output buffers in the I/O blocks provide 4-mA drive for high fan-out CMOS- or TTL-compatible signal levels.



LCA Internal Structure

PART NUMBER	LOGIC CAPACITY (USABLE GATES)	CONFIG-URABLE LOGIC BLOCKS	USER I/Os	CONFIG-URATION PROGRAM (BITS)
M2064	1200	64	58	12038
M2018	1800	100	74	17878

Configurable Logic Block

An array of Configurable Logic Blocks (CLBs) provides the functional elements from which the user's logic is constructed. The Logic Blocks are arranged in a matrix in the center of the device. The M2064 has 64 such blocks arranged in an 8-row by 8-column matrix. The M2018 has 100 logic blocks arranged in a 10 by 10 matrix.

Each logic block has a combinatorial logic section, a storage element, and an internal routing and control section as shown in Figure 2. Each CLB has four general-purpose inputs: A, B, C, and D; and a special clock input (K), which may be driven from the interconnect adjacent to the block. Each CLB also has two outputs, X and Y, which may drive interconnect networks.

Additional memory bits are used to set the user-definable path selectors, shown in Figure 2, which determine CLB internal connections. All memory bits are determined automatically by the XACT design editor as the design is entered.

The logic block combinatorial logic uses a table look-up memory to implement Boolean functions. This technique can generate any logic function of up to four variables with a high-speed, sixteen-bit memory. The propagation delay through the combinatorial network is independent of the function generated. Each block can perform any function of four input variables or any two functions of three input variables each. The input variables may be selected from among the four inputs and the block's storage element output "Q."

Programmable Interconnect

Programmable interconnection resources in the Logic Cell Array provide routing paths to connect inputs and outputs of the I/O and logic blocks into desired networks. All interconnections are composed of metal segments, with programmable switching points provided to implement the necessary routing. Three types of resources accommodate different types of networks:

- General-purpose interconnect
- Long lines
- Direct connection

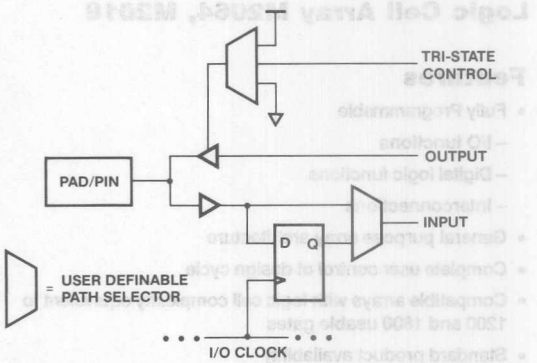


Figure 1. IOB Logic Equivalent

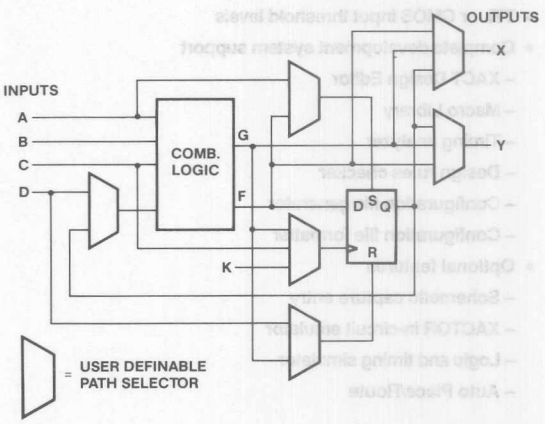


Figure 2. CLB Logic Equivalent

Summary of CLB Switching Characteristics

SYMBOL	PARAMETER		SPEED GRADE						UNIT
			-33		-50		-70		
			MIN	MAX	MIN	MAX	MIN	MAX	
t_{ILO}	Logic input to output	Combinatorial		20		15		10	ns
t_{CKO}	K Clock	To output		20		15		10	ns
t_{ICK}		Logic-input setup	12		8		7		
t_{CKI}		Logic-input hold	0		0		0		
t_{PID}	Input/Output	Pad to input (direct)		12		8		6	ns
t_{OP}		Output to pad (enabled)		15		12		9	ns
F_{CLK}	Maximum flip-flop toggle frequency		33		50		70		MHz

LCA-MDS21 XACT Design Editor System

Features

- Runs on an IBM® PC-XT™ or compatible computer
- Complete basic system for designing with Logic Cell Arrays
- Interactive graphical design editor
- Simplified definition, placement and interconnection capability for logic design and implementation
- Macro library of 113 standard logic family equivalents
- Utility for user-defined macros
- Boolean equation or Karnaugh map alternatives to specify logic functions
- Point-to-point timing calculations for critical path analysis
- Automatic design consistency checking for connectivity and design violations
- Documentation support with hardcopy output of logic and physical configuration information
- Download cable to transfer configuration programs from personal computer to LCA in target system
- Compatible hardware and software options to enhance design productivity
- File formatter for EPROM programmer

General

The XACT Design Editor provides users with a complete design and development system for specification and implementation of designs using Monolithic Memories' Logic Cell Arrays. Functional definition of Configurable Logic Blocks (CLBs), Input/Output Blocks (IOBs) and interconnection is performed with a menu-driven interactive graphics editor. An automatic router greatly reduces the effort to interconnect logic.

Designs are captured with a graphics-based design editor using either a mouse for menu-driven entry, or a keyboard for command-driven entry. Functions are specified by CLB and IOB definitions plus their interconnections. The macro library and user-defined macros enable the user to easily implement complex functions.

The check for logic connectivity and design rule violation is easily performed. All unused internal nodes are automatically configured to minimize power dissipation.

Interactive point-to-point timing delay calculation is provided for timing analysis and critical path determination. This ability enables the user to quickly identify and correct timing problems while the design is in progress.

Automatic generation of similar input netlist files with timing parameters simplifies the use of P-SILOS for logic and timing simulation.

The XACT Design Editor includes hardcopy generation to document a design and automatically track design changes. Logic Cell Array configuration programs can be automatically translated into standard EPROM programming bit pattern formats.

A download cable included with XACT is useful for transferring configuration programs serially from the PC workstation to a Logic Cell Array installed in a system. During product development and debug this capability can be used to save the time required to write a modified configuration program into an EPROM.

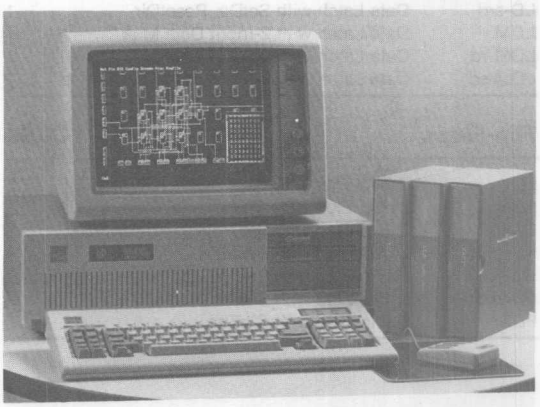
Monolithic Memories provides ongoing support for XACT users. For the first year, software updates are included. After that the user may purchase the LCA-MSC21 Annual Support Agreement to continue to receive the latest software releases. XACT users also receive Monolithic Memories' technical information, which includes information about Logic Cell Arrays and PAL® devices, as well as software updates and application notes for designers. In addition, Monolithic Memories provides comprehensive field and factory support.

System Requirements

Minimum System Configuration

IBM PC-XT, PC-AT or compatible computer with:

- MS-DOS™ 2.1 or higher
- 1M Bytes RAM
- 1 Diskette Drive
- 10-MB Hard Disk
- IBM compatible Color Graphic Adapter and Display
- 1 Serial Interface Port
- 1 Parallel Interface Port
- Mouse System™, Microsoft® or compatible mouse



Design Editor with Routed Design

XACT Macro Library

General		CLBs
GADD	Adder	1
GCOMP	Compare	1
GEQGT	Equal or Greater	1
GMAJ	Majority	1
GMux	2-to-1 Mux	1
GPAR	Parity	1
GXOR	Exclusive-OR	1
GXOR2	Dual Exclusive-OR	1
GXTL	Crystal Oscillator	0 + 2IOB
GOSC	Low Frequency Resistor-Capacitor Oscilator	1 + 2IOB

Pads		IOBs
PIN	Input Pad	1
PINQ	Input Pad with Storage	1
PIO	Input/Output Pad	1
PIOQ	Input/Output Pad with Input Storage	1
PIOC	Input/Output Pad with 'Open Collector'	1
PIOQC	Input/Output Pad with Input Storage, 'Open Collector'	1
POUT	Output Pad	1
POUTC	Output Pad with 'Open Collector'	1
POUTZ	Output Pad with 3-State Control	1
PREG	Output Pad with Input Storage	1

Latches		CLBs
LD	Data Latch	1
LC-rd	Data Latch with ResetDir	1
LC-sd	Data Latch with SetDir	1
LD-srd	Data Latch with SetDir, ResetDir	1
LDM	Data Latch with 2-Input Data Mux	1
LDM-rd	Data Latch with 2-Input Data Mux, ResetDir	1
LDM-sd	Data Latch with 2-Input Data Mux, SetDir	1

Flip-Flops		CLBs
FD	D Flip-Flop	1
FDR	D Flip-Flop with Reset	1
FDS	D Flip-Flop with Set	1
FD-rd	D Flip-Flop with ResetDir	1
FD-sd	D Flip-Flop with SetDir	1
FD-srd	D Flip-Flop with SetDir, ResetDir	1
FDC	D Flip-Flop with ClkEna	1
FDCR	D Flip-Flop with ClkEna, Reset	1
FDCS	D Flip-Flop with ClkEna, Set	1
FDM	D Flip-Flop 2-Input Data Mux	1
FDMR	D Flip-Flop 2-Input Data Mux, Reset	1
FDMS	D Flip-Flop 2-Input Data Mux, Set	1
FDM-rd	D Flip-Flop 2-Input Data Mux, ResetDir	1
FDM-sd	D Flip-Flop 2-Input Data Mux, SetDir	1
FSR	Set-Reset Flip-Flop with Set Dominate	1
FRS	Set-Reset Flip-Flop with Reset Dominate	1
FJK	J-K Flip Flop	1
FJKS	J-K Flip Flop with Synchronous Set	1
FJK-rd	J-K (Set-Reset) Flip Flop with ResetDir	1
FJK-sd	J-K (Set-Reset) Flip Flop with SetDir	1
FJK-srd	J-K (Set-Reset) Flip Flop with SetDir, ResetDir	1
FT0	Self Toggle Flip-Flop	1
FT0R	Self Toggle Flip-Flop with Reset	1
FT	Toggle Flip-Flop	1
FTP	Toggle Flip-Flop with ParEna	1
FTP-rd	Toggle Flip-Flop with ParEna, ResetDir	1
FTR	Toggle Flip-Flop with Reset	1
FTS	Toggle Flip-Flop with Set	1
FT2	2-Input Toggle Flip-Flop	1
FT2R	2-Input Toggle Flip-Flop with Reset	1

Decoders		CLBs
D2-4	1-of-4 Decoder	2
D2-4E	1-of-4 Decoder, with Ena	2
74-139	1-of-4 Single Decoder with Low Output, Ena	4
D3-8	1-of-8 Decoder	5
D3-8E	1-of-8 Decoder with Ena	6
74-138	1-of-8 Decoder with Enables, Low Output	7
74-42	1-of-10 Decoder with Low Output	8

Multiplexers		CLBs
M3-1	3-to-1 Mux	2
M3-1E	3-to-1 Mux with Ena	2
M4-1	4-to-1 Mux	3
M4-1E	4-to-1 Mux with Ena	3
74-352	4-to-1 Mux with Low Output, Ena	3
M8-1	8-to-1 Mux	7
M8-1E	8-to-1 Mux with Ena	7
74-151	8-to-1 Mux with Ena Complementary Outputs	7
74-152	8-to-1 Mux with Low Output	7

XACT Macro Library

Registers

		CLBs
Data Registers		
RD4	4-Bit Data Register	4
RD8	8-Bit Data Register	8
RE8CR	8-Bit Data Register with ClkEna, Reset	8
Serial to Parallel		
RS4	4-Bit Shift Register	4
74-195	4-Bit Serial to Parallel Shift Register with ParEna, Reset	5
74-194	4-Bit Bidirectional Shift Register with ClkEna, ParEna, ResetDir	12
RS8	8-Bit Shift Register	8
RS8CR	8-Bit Shift Register with ClkEna, Reset	8
RS8PR	8-Bit Shift Register with ParEna, Reset	8
RS8R	8-Bit Shift Register with Reset	8
74-164	8-Bit Serial to Parallel Shift Register with ResetDir	8

Counters

		CLBs
Modulo 2		
C2BCR	1-Bit Binary Counters with ClkEna, Reset	1
C2BC-rd	1-Bit Binary Counters with ClkEna, ResetDir	1
C2BP	1-Bit Binary Counters with ParEna	1
C2BR	1-Bit Binary Counters with Reset	1
C2B-rd	1-Bit Binary Counters with ResetDir	1
Modulo 4		
C4BCP	2-Bit Binary Counters with ClkEna, ParEna	3
C4BCR	2-Bit Binary Counters with ClkEna, Reset	2
C4BC-rd	2-Bit Binary Counters with ClkEna, ResetDir	2
C4JCR	2-Bit Johnson Counters with ClkEna, Reset	2
Modulo 6		
C6JCR	3-Bit Johnson Counter with ClkEna, Reset	3
Modulo 8		
C8BCP	3-Bit Binary Counters with ClkEna, ParEna	5
C8BCR	3-Bit Binary Counters with ClkEna, Reset	4
C8BC-rd	3-Bit Binary Counters with ClkEna, ResetDir	4
C8JCR	3-Bit Johnson Counters with ClkEna, Reset	4
Modulo 10		
C10BC-rd	4-Bit BCD Counter with ClkEna, ResetDir	4
C10BCP-rd	4-Bit BCD Counter with ClkEna, ParEna, ResetDir	7
74-160	4-Bit BCD Counter with ClkEna, ParEna, ResetDir	8
C10BP-rd	4-Bit BCD Counter with ParEna, ResetDir	6
C10JCR	5-Bit Johnson Counter with ClkEna, Reset	5
Modulo 12		
C12JCR	6-Bit Johnson Counter with ClkEna, Reset	6
Modulo 16		
C16BA-rd	4-Bit Binary Ripple Counter with ResetDir	4
C16BC-rd	4-Bit Binary Counter with ClkEna, ResetDir	4
C16BCPR	4-Bit Binary Counter with ClkEna, ParEna, Reset	10
C16BCP-rd	4-Bit Binary Counter with ClkEna, ParEna, ResetDir	6
74-161	4-Bit Binary Counter with ResetDir	8
C16BP-rd	4-Bit Binary Counter with ParEna, ResetDir	5
C16BUD-rd	4-Bit Binary Up-Down Counter with ParEna, ResetDir	8
C16JCR	8-Bit Johnson Counter with ClkEna, Reset	8
Modulo 256		
C256FC-rd	8-Bit Modulo 256 Feedback Shift Register with ClkEna, ResetDir	9

Features

- **Event-driven logic and timing simulator**
- **Logic network input automatically generated by XACT Design Editor**
- **Control and observation of any physical circuit node**
- **Multiple file input for vectors and commands**
- **Interactive or batch mode operation**
- **Output available in printed or tabular formats**
- **Runs on an IBM PC-XT, PC-AT or compatible personal computer**

General

P-SILOS is a powerful PC-based simulator that provides event-driven logic and timing simulation of Logic Cell Array designs. Simulation is particularly useful for testing logic or logic segments as well as for verifying critical timing over worst case power supply, temperature and process conditions.

Simulation is useful in several stages of the design cycle. After design entry, simulation may be used to debug logic in an unplaced and unrouted design. This saves design time because logic errors can be detected and corrected prior to final placement and routing. After a circuit has been placed, routed, and then fully debugged using in-circuit emulation, worst case timing may be verified. This enables the user to select the correct Logic Cell Array speed for a particular application.

Network inputs for Logic Cell Array designs are automatically created by the Simgen utility in the XACT system. The network includes logic and routing delay parameters and setup and hold times based upon the selected speed grade operating under worst case conditions. Simulation stimuli are created with a set of clock statements or with an input pattern for either pad inputs or internal nodes. Simulation results are available in tabular, plotted, and graphic formats. This flexibility makes debugging easy for both the circuit function and timing.

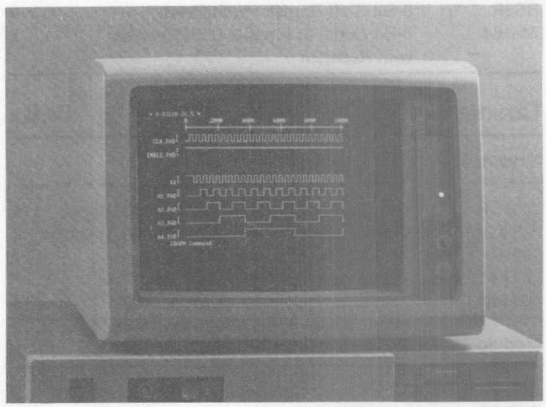
System Requirements

Minimum System Configuration

IBM PC-XT, PC-AT or compatible computer with:

- MS-DOS 2.1 or higher
- 640 K Bytes RAM
- 1 Diskette Drive
- 10-MB Hard Disk
- 1 Parallel Interface Port

Refer to the MDS21 XACT Design Editor Product Datasheet for additional equipment required for systems which will also run the XACT Design Editor.



P-SILOS Waveform Output

LCA-MDS23 Automatic Placement and Routing Program

Features

- **Automatic placement and routing of logic to minimize design cycle time**
- **User control over placement of logic blocks**
- **User specification of critical paths**
- **Netlist inputs from either schematic capture or XACT**
- **May be used in conjunction with schematic capture or with the XACT Design Editor**
- **Runs on IBM PC-XT, PC-AT or compatible personal computer**

General

The automatic Placement and Routing program enhances the productivity of designers using Logic Cell Arrays by reducing design placement and routing time, whether the design logic is entered from a schematic capture package or from the XACT Design Editor.

Designs that are developed incrementally can also take advantage of Automatic Placement and Routing. Partial Logic Cell Array layouts can be locked in place while additions to the design are automatically placed and routed, or the design can be completely rearranged to yield a new placement.

The Automatic Placement and Routing program is extremely flexible. Through placement directives the user can control the placement process to achieve the best placement for a particular design. Routing resources can be specified to minimize clock skews and signal delays for critical paths. The result is faster product developement.

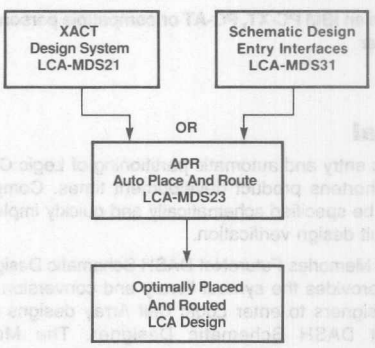
System Requirements

Minimum System Configuration

IBM PC-XT, PC-AT or compatible computer with:

- MS-DOS 2.1 or higher
- 640 K Bytes RAM
- 1 Diskette Drive
- 10-MB Hard Disk
- 1 Parallel Interface Port

Refer to the MDS21 XACT Design Editor Product Datasheet for additional equipment required for systems which will also run the XACT Design Editor.



APR Diagram

2

LCA-MDS31 FutureNet DASH Schematic Design Entry Interface

Features

- **Design entry to XACT via the FutureNet DASH Schematic Designer**
- **Macro library of over 100 standard logic family equivalents derived from the XACT Macro Library**
- **Library of logic symbols including all two-input, three-input, and four-input AND, OR, and XOR gates plus storage, input/output, and clock elements**
- **User control for flagging critical paths for the LCA-MDS23 Automatic Placement and Routing Program**
- **Automatic partitioning and conversion of schematic drawings to a Monolithic Memories' Logic Cell Array design file**
- **Output compatibility with XACT Design Editor and the Automatic Placement and Routing Program**
- **Runs on an IBM PC-XT, PC-AT or compatible personal computer**

General

Schematic entry and automatic partitioning of Logic Cell Array designs shortens product development times. Complex designs can be specified schematically and quickly implemented for in-circuit design verification.

Monolithic Memories FutureNet DASH Schematic Design Entry Interface provides the symbol library and conversion utility to permit designers to enter Logic Cell Array designs with the FutureNet DASH Schematic Designer. The Monolithic Memories module provides the logic, I/O and macro symbols to be used in the schematic and a conversion utility which automatically partitions and translates the schematic into a Logic Cell Array design.

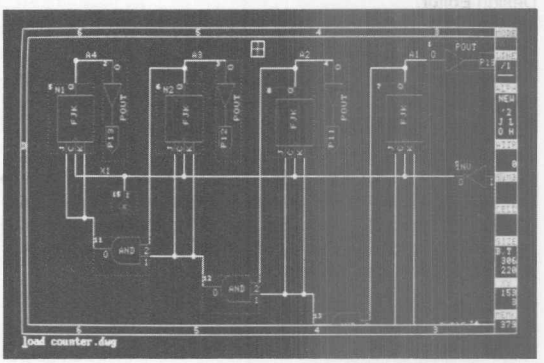
System Requirements

Minimum System Configuration

IBM PC-XT, PC-AT or compatible computer with:

- FutureNet DASH-2 or later, and associated hardware including mouse, Enhanced Graphics Adapter and Display
- MS-DOS 2.1 or higher
- 640 K Bytes RAM
- 1 Diskette Drive
- 10-MB Hard Disk

Refer to the MDS21 XACT Design Editor Product Datasheet for additional equipment required for systems which will also run the XACT Design Editor.



Schematic Capture

LCA-MDS24, LCA-MDS26, LCA-MDS27 XACTOR In-Circuit Emulator

Features

- Real-time in-circuit emulation in user's target system
- Concurrent emulation of up to four devices
- Readback and display of Logic Cell Array internal storage element states
- Device status display with automatic update of asynchronous events
- Control and I/O pin isolation from target system
- Support for daisy chain programming of up to seven devices in a daisy chain
- On-chip crystal oscillator support during emulation
- Support for multiple device and package types
- Runs on an IBM PC-XT, PC-AT or compatible personal computer

General

The XACTOR real-time in-circuit emulator provides interactive target-system emulation of up to four Logic Cell Arrays from the host PC system. In-circuit emulation provides a powerful productivity enhancement to simulation, providing capabilities to verify functionality in the target system at full speed with all other circuits and system software.

The emulation system is composed of a microcomputer-based controller (LCA-MDS24), and from one to four universal emulation pods (LCA-MDS26), each with a package-specific emulation header (LCA-MDS27). One universal emulation pod is included with the system. The controller is connected to the host PC through a serial port and provides local storage of configuration programs, control of individual device configurations and control of the isolation of the pod device(s) from the target system. The user can set the state and isolation for each of the control signals to provide debugging of target hardware. Four general I/O pins are available to provide test points which may also be isolated from the target system.

Target Logic Cell Arrays can be programmed individually or in a daisy chain. Daisy chains of up to seven devices may be supported from any of the four pods. Individual device isolation and configuration is controlled with mouse or keyboard commands and may be supplemented with user-defined setup files for easy system debugging.

Readback of device configuration may be performed on command for verification of the configuration process and interrogation of the internal states. The state of all internal storage elements is displayed after readback has been performed. Status displays showing the state of all isolation switches and control signal states are provided. The status display includes automatic reporting of asynchronous status changes in the target system.

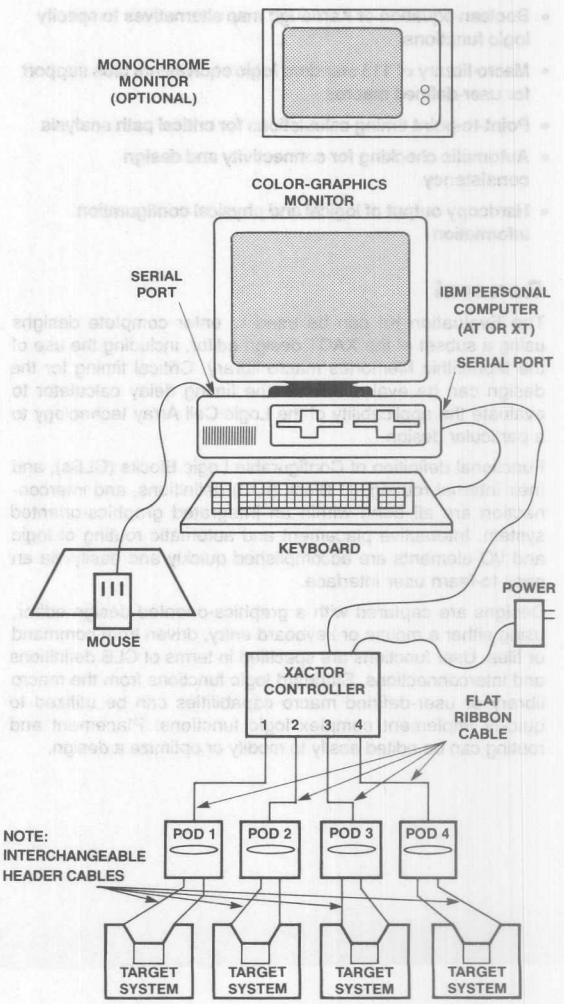
Universal In-Circuit Emulator Pod (LCA-MDS26)

Additional pods may be connected to the XACTOR in-circuit emulator controller, up to a maximum of four pods per controller. Pod headers (LCA-MDS27) are interchangeable for different device and package types. Each pod provides a direct in-socket connection for a minimum disruption of the target system. Test points are provided to allow connection of a logic analyzer or other test equipment to aid in the system debugging.

System Requirements

Minimum System Configuration

IBM PC-XT, PC-AT or compatible computer as configured for MDS21 XACT Design Editor, plus second serial interface port.



XACTOR Hardware

LCA-MEK01 Logic Cell Array Evaluation Kit

The Monolithic Memories Logic Cell Array is a high-performance CMOS user-programmable gate array. The Monolithic Memories' Logic Cell Array Evaluation Kit is a software package that provides the capability to evaluate the Logic Cell Array for new applications.

Features

- **Design software package for IBM PC-XT, PC-AT or compatible computer**
- **Interactive graphics-oriented designer interface**
- **Simplified definition, placement and connection capability for implementation of complex logic**
- **Boolean equation or Karnaugh map alternatives to specify logic functions**
- **Macro library of 113 standard logic equivalents plus support for user-defined macros**
- **Point-to-point timing calculations for critical path analysis**
- **Automatic checking for connectivity and design consistency**
- **Hardcopy output of logical and physical configuration information**

General

The Evaluation Kit can be used to enter complete designs using a subset of the XACT design editor, including the use of the Monolithic Memories macro library. Critical timing for the design can be evaluated with the timing delay calculator to evaluate the applicability of the Logic Cell Array technology to a particular design.

Functional definition of Configurable Logic Blocks (CLBs), and their internal routing,I/O Block (IOB) definitions, and interconnection are all done within an integrated graphics-oriented system. Interactive placement and automatic routing of logic and I/O elements are accomplished quickly and easily via an easy-to-learn user interface.

Designs are captured with a graphics-oriented design editor, using either a mouse or keyboard entry, driven from command or files. User functions are specified in terms of CLB definitions and interconnections. Standard logic functions from the macro library or user-defined macro capabilities can be utilized to quickly implement complex logic functions. Placement and routing can be edited easily to modify or optimize a design.

Checking of logical connectivity is performed automatically. All unused internal nodes are automatically configured to minimize power dissipation.

Interactive point-to-point timing delay calculation is provided to simplify timing analysis and critical path determination.

The Evaluation Kit includes hardcopy generation to document a design and automatically track design changes.

System Requirements

Minimum System Configuration

IBM PC-XT, PC-AT or compatible computer with:

- MS-DOS 2.1 or higher
- 640K Bytes RAM
- 1 Diskette Drive
- 10-MB Hard Disk
- IBM or compatible Color Graphic Adapter and Display
- 1 Serial Interface Port
- Mouse Systems, Microsoft or compatible mouse

Evaluation Kit

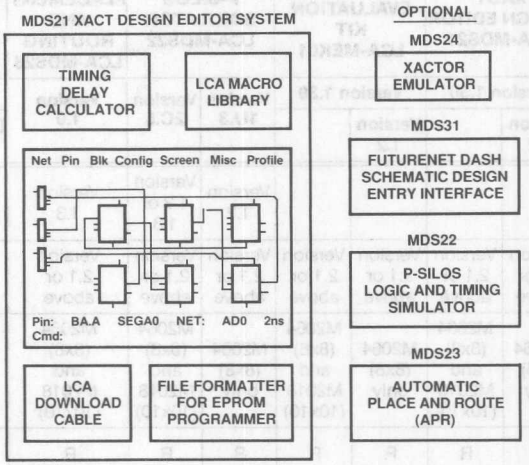
Minimum Requirements of Software and Hardware Configurations for Monolithic Memories LCA Design System

Legend R Required S Supported — Not required or supported		Software Package											
		XACT DESIGN EDITOR LCA-MDS21		XACT EVALUATION KIT LCA-MEK01		P-SILOS SIMULATOR LCA-MDS22		AUTOMATIC PLACEMENT AND ROUTING LCA-MDS23	FutureNet DASH SCHEMATIC DESIGN ENTRY INTERFACE LCA-MDS31			XACTOR IN-CIRCUIT EMULATOR LCA-MDS24	
		Version 1.30		Version 1.30		Version 1U.3	Version 2C.5	Version 1.0	Version 1.00**			1.10	1.30 1.33*
		Version 1.2		Version 1.2					DASH 2	DASH 3C	DASH 4		
XACT Version Required						Version 1.2	Version 1.2 or 1.3	Version 1.3	Version 1.3			Version 1.2	Version 1.3
MS-DOS PC-DOS Operating System		Version 2.1 or above	Version 2.1 or above	Version 2.1 or above	Version 2.1 or above	Version 2.1 or above	Version 2.1 or above	Version 2.1 or above	Version 2.1 or above			Version 2.1 or above	Version 2.1 or above
Logic Cell Arrays Supported		M2064 (8x8) only	M2064 (8x8) and M2018 (10x10)	M2064 (8x8) only	M2064 (8x8) and M2018 (10x10)	M2064 (8x8) only	M2064 (8x8) and M2018 (10x10)	M2064 (8x8) and M2018 (10x10)	M2064 (8x8) and M2018 (10x10)			M2064 (8x8) 68NL only	M2064* (8x8) and M2018 (10x10)
IBM PC XT or 100% compatible		R	R	R	R	R	R	R	R			R	R
IBM PC AT or 100% compatible		S	S	S	S	S	S	S	S			S	S
Memory	Minimum System RAM	640 KB	1 MB	640 KB	1 MB	640 KB	640 KB	640 KB	256 KB	512 KB	512 KB	640 KB	1 MB
	Hard Disk (10 MB min 30 MB REC)	R	R	R	R	R	R	R	R	R	R	R	R
Graphics Boards and Displays	Monochrome	—	—	—	—	R	R	—	R	—	—	—	—
	CGA (Color graphics adapter)	R	R	R	R	S	S	R	—	—	—	R	R
	EGA (Enhanced color graphics adapter)	S	S	S	S	S	S	S	—	R (with 192 KB)	R (with 192 KB)	S	S
	Lotus/Intel EMS (Expanded memory specifications)	—	R (with 256 KB)	—	R (with 256 KB)	—	—	—	—	—	—	—	R (with 256 KB)
	Vendor graphics board								Future-Net graphics controller board				
Other Devices	Security key	R	R	—	—	R	R	R	—	—	—	R	R
	Mouse	R†	R†	R†	R†	—	—	—	R (Future-Net)	R (Future-Net)	R (Future-Net)	R†	R†
	Min. number of parallel ports	1	1	1	1	1	1	1	1	Future-Net mouse/ parallel port board	Future-Net mouse/ parallel port board	1	1
	Min. number of serial ports	1	1	1	1	0	0	0	0	0	0	2	2

* XACTOR Version 1.33 supports the universal emulator pod with interchangeable header cables for each package type. Versions 1.10 and 1.30 or XACTOR support only the dedicated 68-Pin PLCC emulation pod originaly offered with XACTOR Version 1.10. XACTOR Version 1.33 will also support the original 68-Pin PLCC Emulator Pod.

** LCA-MDS31 FutureNet DASH Schematic Design Entry Interface version 1.00 is compatible with FutureNet DASH Schematic Designer versions 2, 3C and 4.

† Must be Mouse Systems™, FutureNet® or Microsoft® mouse compatible.



Development System

The DS21 XACT Design Editor provides all capabilities required for Logic Cell Array design. Additional development system options provide enhanced designer productivity during design entry, placement and routing, and design verification.

Military CMOS Programmable Gate Array Logic Cell™ Array M2064

Conforms to MIL-STD-883, Class B*

US Patent 4124899

Features

CMOS
- Low power
- TTL or CMOS input threshold levels

PROGRAMABLE
- Programmable Logic functions
- Programmable I/O blocks
- Programmable interconnects

STATIC RAM BASED
- Reprogrammable
- Reconfigurable

SECURITY
- User selectable Security Mode
- Verification feature

Benefits

- Reduced power supply
- Higher board densities
- Complete user control of design
- Instant prototyping
- Replaces SSI and MSI devices
- Easy design modification
- Selectable configuration modes
- 100% testable
- Protects proprietary designs
- Eases design debug

Ordering Information

M2064 - 50 M P /883, B

PART NUMBER
M2064 (1200 Gates, 58 I/OBs)

SPEED GRADE
-20 (20 MHz Toggle Rate
-33 (33 MHz Toggle Rate)
-50 (50 MHz Toggle Rate)

TEMPERATURE RANGE
M = −55°C to + 125°C

PROCESSING
883,B = MIL-STD-883, Class B

PACKAGING
P = Pin Grid Array (68 pin)
L = Leadless Chip Carrier (68 pin)

Description

The Military CMOS Logic Cell Array bridges the gap between Programmable Logic Devices (PLDs) and gate arrays. This high-density, low-power Logic Cell Array device provides designers with both the density benefits of gate arrays and the programmability benefits of user-configurable devices.

The flexible architecture of the LCA is similar to that of a gate array, with an interior matrix of programmable logic blocks called Configurable Logic Blocks (CLBs), a surrounding ring of programmable I/O blocks (IOBs) and programmable interconnects used to define the overall device structure. Unlike gate arrays, LCA functionality is user defined by loading the internal writable storage cells with the configuration data. The reprogrammability of the SRAM based LCA allows instant design modification on the bench and on the board. Due to the SRAM based architecture of the LCA, the radiation tolerance data for the M2064 is similar to industry SRAMs which disply Total Dose levels from 10^4 to 10^6 rads (Si).

Applications for the LCA cover a wide spectrum of uses. With its high gate density and low-power CMOS technology, the LCA is an ideal low-cost gate array good for prototyping as well as production. The LCA's SRAM-based architecture allows it to be used in applications that take advantage of its reconfigurability. Ground systems (radar) can use the LCA as reconfigurable hardware replacing several devices and saving board space. In microprocessor systems the LCA can be used as system reconfigurable logic blocks. In the artificial intelligence arena, the "fuzzy logic" of the LCA provides the designer with added flexibility. The SRAM-based logic of the LCA allows for pattern security of sensitive designs when the device is removed from its board. When a different mode backup device is needed the LCA provides excellent redundancy. In short the LCA offers the total ASIC solution.

Silicon Menu

MMI PART*	ORGANIZATION	EQUIVALENT GATE COUNT	CONFIGURABLE LOGIC BLOCKS	USER I/Os	CONFIGURATION PROGRAM BITS	MAX STANDBY CURRENT (CMOS INPUTS)	MAX STANDBY CURRENT (TTL INPUTS)	PACKAGES	MAX TOGGLE RATE BETWEEN CLBs
M2064-20	8x8	1200	64	58	12040	5 mA	10 mA	68LCC, 68PGA	20 MHz
M2064-33	8x8	1200	64	58	12040	5 mA	10 mA	68LCC, 68PGA	33 MHz

* M2064-50 Contact Factory.

2175 Mission College Blvd. Santa Clara, CA 95054-1592 Tel: (408) 970-9700 TWX: 910-338-2376 TWX: 910-338-2374

Monolithic Memories MMI

Absolute Maximum Ratings

Supply voltage, V_{CC} ..... −0.5 V to 7.0 V
Power down, V_{CC} ..... 2.0 V to 7.0 V
Input voltage range ..... −1.5 V to 5.5 V
Voltage applied to three-state output ..... −0.5 V to 5.5 V
Storage temperature ..... −65°C to +150°C
Terminal temperature, Leadless Chip Carrier package (Soldering, 10 seconds) ..... 240°C
Thermal resistance, junction to case, θ_{jcmax}, and junction to ambient, θ_{jamax}

Package	θ_{jcmax}	θ_{jamax} (Still air)
(L) Leadless Chip Carrier	1.5°C/W	32°C/W
(P) Pin Grid Array	3°C/W	45°C/W

Maximum power dissipation ..... See Commercial datasheet
Maximum junction temperature ..... 175°C
Maximum current density ..... Contact factory

Software/Hardware Menu

MMI PARTS	
LCA-MDS21	LCA Development System
LCA-MSC21	LCA Development System Annual Support Agreement
LCA-MDS22	LCA Simulator (P-Silos™)
LCA-MDS23	LCA Automatic Placement and Routing Program

MMI PARTS	
LCA-MDS24	LCA In-Circuit Emulator
LCA-MDS28	LCA Universal Pod
LCA-MDS27XX	LCA Package Specific Pod Headers
LCA-MDS31	LCA Futurenet® Interface
LCA-MDS33	LCA Daisy Interface
LCA-MEK01	LCA Evaluation Kit

3

Operating Conditions

SYMBOL	PARAMETER	MIN	TYP	MAX	UNIT
V_{CC}	Supply voltage relative to GND	4.5		5.5	V
V_{IHT}	High level input voltage—TTL configuration	2.0		V_{CC}	V
V_{IHC}	High level input voltage—CMOS configuration	0.7 V_{CC}		V_{CC}	V
V_{ILT}	Low level input voltage—TTL configuration	0		0.8	V
V_{ILC}	Low level input voltage—CMOS configuration	0		0.2 V_{CC}	V
I_{IT}	Input leakage current—TTL configuration	±1			μA
I_{IC}	Input leakage current—CMOS configuration	±1			μA
I_{OZ}	Three-state output off current (V_{CC} = 5.5 V)	±10			μA
t_{OP}	Operating free-air temperature	−55		+125	°C

Electrical Characteristics Over Operating Conditions

Conforms to MIL-STD-883 Group A Subgroups 1, 2 and 3

SYMBOL	PARAMETER		TEST CONDITION		MIN	TYP	MAX	UNIT
V_{OH}	High level output voltage		V_{CC} = MIN	I_{OH} = −4.0 mA	3.7			V
V_{OL}	Low level output voltage		V_{CC} = MIN	I_{OL} = 4.0 mA			0.4	V
I_{CCO}	Quiescent operating power supply current	CMOS inputs	V_{CC} = 5.0 V				5	mA
		TTL inputs	V_{CC} = 5.0 V				10	mA
I_{CCPD}	Power down supply current		V_{CC} = 2.0 V				0.5	mA

Power On Timing

The LCAs contain on-chip reset timing logic for power-up operation. To insure proper master mode system operation, VCC must rise from 2.0 V to minimum specification level in 10 ms or less. For other modes, initiation of configuration must be delayed for 60 ms after VCC reaches the minimum specified level.

Test Conditions

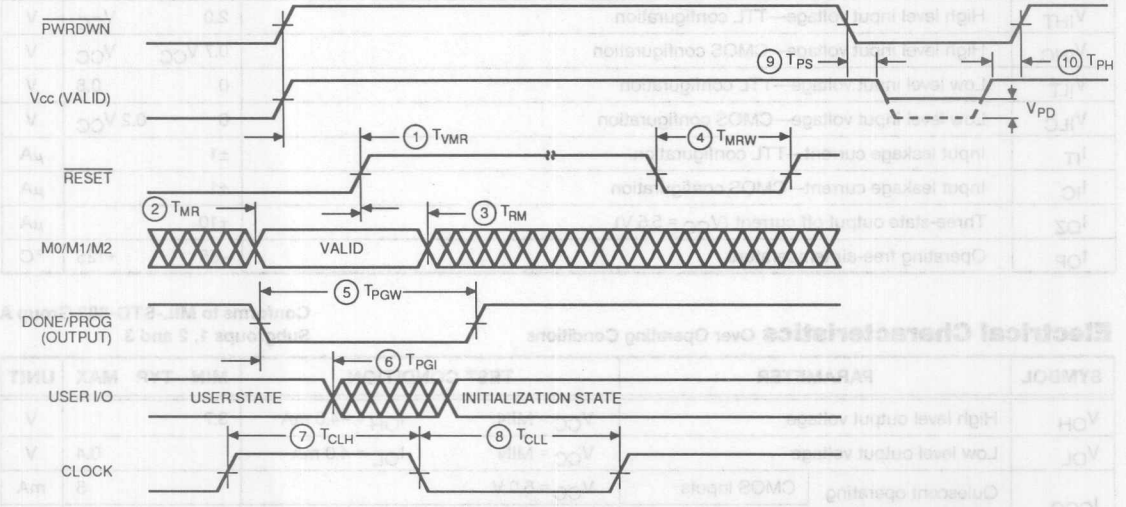
Outputs loaded with rated DC current and 50-pF capacitance to GND.

Switching Characteristics — General

Conforms to MIL-STD-883 Group A Subgroups 9, 10 and 11*

SYMBOL	DESCRIPTION		-20		-33		UNIT
			MIN	MAX	MIN	MAX	
t_{VMR} ①	$\overline{RESET}$ (2)	V_{CC} setup (2.0 V)	250		150		ns
t_{MR} ②		M2, M1, M0 setup	100		60		ns
t_{RM} ③		M2, M1, M0 hold	100		60		ns
t_{MRW} ④		Width (LOW)	250		150		ns
t_{PGW} ⑤	DONE/$\overline{PROG}$	Program width (LOW)	6		6		µs
t_{PGI} ⑥		Initialization		7		7	µs
t_{CLH} ⑦	CLOCK	Clock (HIGH)	20		12		ns
t_{CLL} ⑧		Clock (LOW)	20		12		ns
t_{PS} ⑨	$\overline{PWR\ DWN}$	Setup to V_{CC}	0		0		ns
t_{PH} ⑩		Hold from V_{CC}	0		0		ns
V_{PD}		Power Down	2.0		2.0		V

Notes: 1. V_{CC} must rise from 2.0 Volts to V_{CC} minimum in lest than 10 ms for master mode.
2. $\overline{RESET}$ timing relative to power-on and valid mode lines (M0, M1, M2) is relevant only when $\overline{RESET}$ is used to delay configuration.
3. Minimum CLOCK widths for the auxiliary buffer are 1.25 times the t_{CLH}, t_{CLL}.
* Contact factory for test macros.



Switching Characteristics — CLB

SYMBOL	DESCRIPTION		-20		-33		UNIT
			MIN	MAX	MIN	MAX	
t_{ILO} ①	Logic input to output	Combinatorial		35		20	ns
t_{ITO} ②		Transparent latch		45		25	ns
t_{QLO}		Additional for Q through F or G to out		30		13	ns
t_{CKO} ⑨	K Clock	To output		35		20	ns
t_{ICK} ③		Logic-input setup	22		12		ns
t_{CKI} ④		Logic-input hold	0		0		ns
t_{CCO} ⑩	C Clock	To output		45		25	ns
t_{ICC} ⑤		Logic-input setup	18		12		ns
t_{CCI} ⑥		Logic-input hold	10		6		ns
t_{CIO} ⑪	Logic input to G Clock	To output		65		37	ns
t_{ICI} ⑦		Logic-input setup	10		6		ns
t_{CII} ⑧		Logic-input hold	15		9		ns
t_{RIO} ⑫	Set/reset direct	Input A or D to out		45		25	ns
t_{RLO} ⑬		Through F or G to out		65		37	ns
t_{MRQ}		Master Reset pin to out		60		35	ns
t_{RS}		Separation of set/reset	30		17		ns
t_{RPW}		Set/reset pulse-width	20		12		ns
F_{CLK}	Flip-flop toggle rate	Q through F to flip-flop	20		33		MHz
t_{CH} ⑭	Clock	Clock HIGH	20		12		ns
t_{CL} ⑮		Clock LOW	20		12		ns

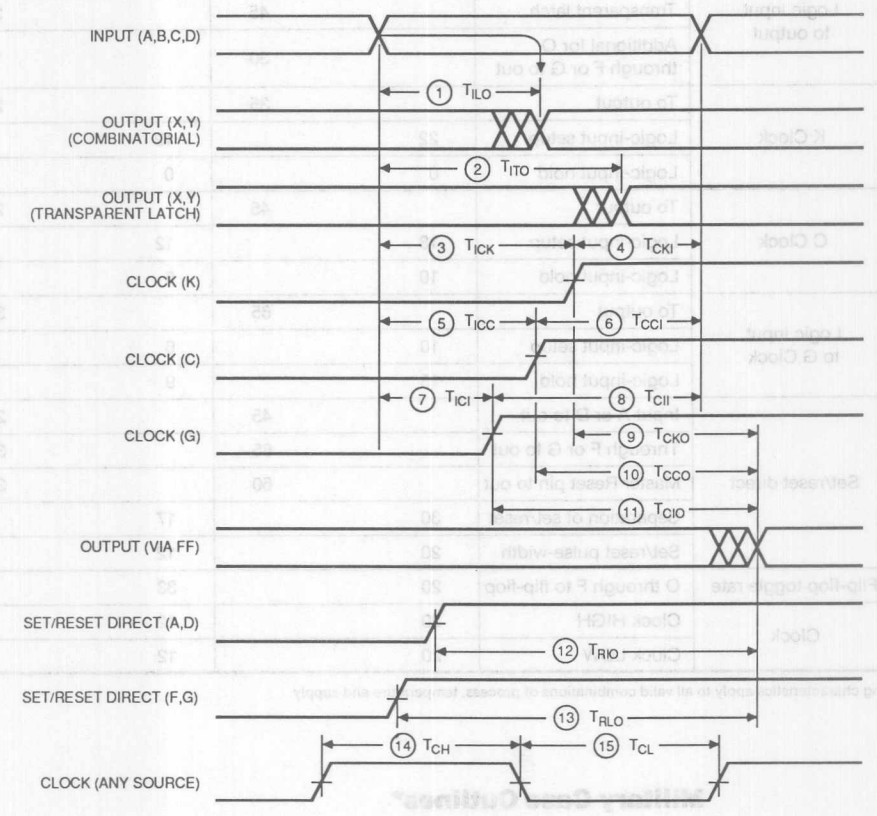
Note: All switching characteristics apply to all valid combinations of process, temperature and supply.

Military Case Outlines*

PACKAGE OUTLINE LETTER	CONFORMS TO MIL-M-38510 APPENDIX C CASE
L	C-7
P	P-BC

* Refer to MIL-M-38510, Appendix C for the appropriate packge drawings.

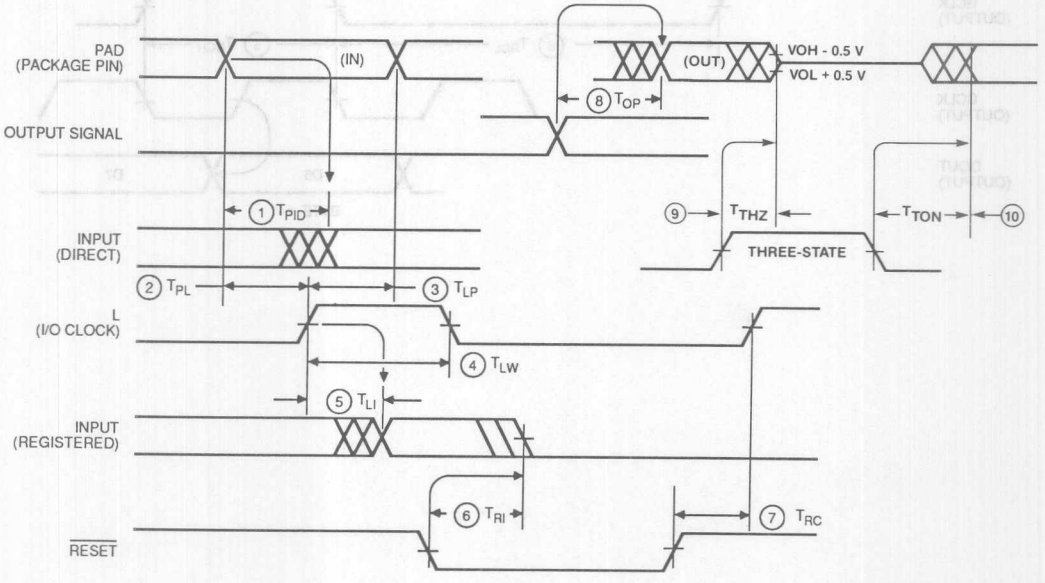
Switching Characteristics CLB



Switching Characteristics — IOB

SYMBOL	DESCRIPTION	-20		-33		UNIT
		MIN	MAX	MIN	MAX	
t_{PID} ①	Pad (package pin) to input (direct)		20		12	ns
t_{LI} ⑤	I/O Clock to input (storage)		30		20	ns
t_{PL} ②	/O Clock to pad-input setup	20			12	ns
t_{LP} ③	I/O Clock to pad-input hold	0		0		ns
t_{LW} ④	I/O Clock pulse width	20		12		nsp
	I/O Clock frequency		20		33	MHz
t_{OP} ⑧	Output to pad (output enabled)		25		15	ns
t_{THZ} ⑨	Three-state to pad begin hi-Z		35		25	ns
t_{TON} ⑩	Three-state to pad end hi-Z		40		25	ns
t_{RI} ⑥	$\overline{\text{RESET}}$ to input (storage)		50		40	ns
t_{RC} ⑦	$\overline{\text{RESET}}$ to input clock		35		35	ns

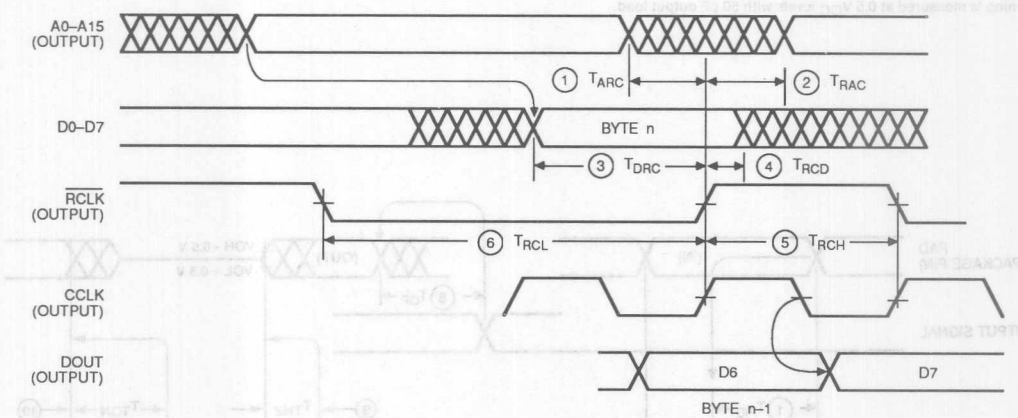
Note: Timing is measured at 0.5 V_{CC} levels with 50 pF output load.



Switching Characteristics — Programming — Master Mode

SYMBOL	DESCRIPTION		-20		-33		UNIT
			MIN	MAX	MIN	MAX	
t_{ARC} ①	RCLK	From address invalid		0		0	ns
t_{RAC} ②		To address valid		300		200	ns
t_{DRC} ③		To data setup	100		60		ns
t_{RCD} ④		To data hold	0		0		ns
t_{RCH} ⑤		RCLK HIGH	600		600		ns
t_{RCL} ⑥		RCLK LOW	4.0		4.0		μs

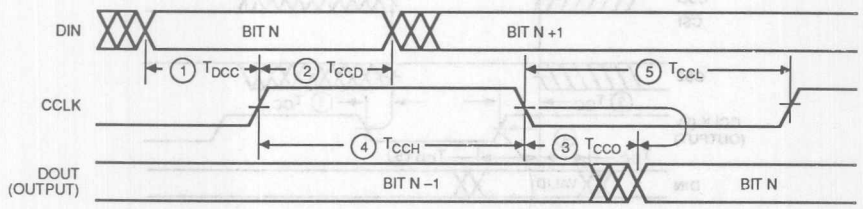
Notes: 1. CCLK and DOUT timing are the same as for slave mode.
2. At power up, V_{CC} must rise from 2.0 V to V_{CC} minimum in less than 10 ms.



Switching Characteristics — Programming — Slave Mode

SYMBOL	DESCRIPTION	-20		-33		UNIT
		MIN	MAX	MIN	MAX	
t_{CCO} ③	CCLK to DOUT		100		65	ns
t_{DCC} ①	CCLK DIN setup	50		25		ns
t_{CCD} ②	CCLK DIN hold	75		40		ns
t_{CCH} ④	CCLK HIGH time	0.50		0.30		µs
t_{CCL} ⑤	CCLK LOW time	0.30	10.0	0.25	5.0	µs
F_{CC}	CCLK frequency		1		2	MHz

Note: Configuration must be delayed at least 40 ms after V_{CC} minimum.

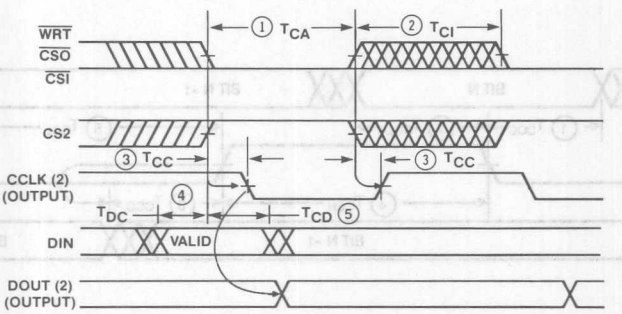


3

Switching Characteristics — Programming — Peripheral Mode

SYMBOL	DESCRIPTION		-20		-33		UNIT
			MIN	MAX	MIN	MAX	
t_{CA} ①	Controls[1] ($\overline{CS0}$, $\overline{CS1}$, CS2, $\overline{WRT}$)	Active (last active input to first inactive)	0.30	10.0	0.20	5.0	μs
t_{CI} ②		Inactive (first inactive input to last active)	0.50		0.30		μs
t_{CCC} ③		CCLK[2]		100		75	ns
t_{DC} ④		DIN setup	50		35		ns
t_{CD} ⑤		DIN hold	10		5		ns

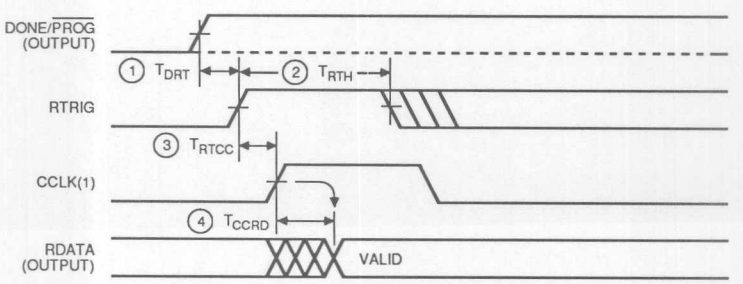
Notes: 1. Peripheral mode timing determined from last control signal of the logical AND of ($\overline{CS0}$, $\overline{CS1}$, CS2, $\overline{WRT}$) to transition to active or inactive state.
2. CCLK and DOUT timing are the same as for slave mode.
3. Configuration must be delayed at least 40 ms after V_{CC} minimum.



Switching Characteristics — Program Readback

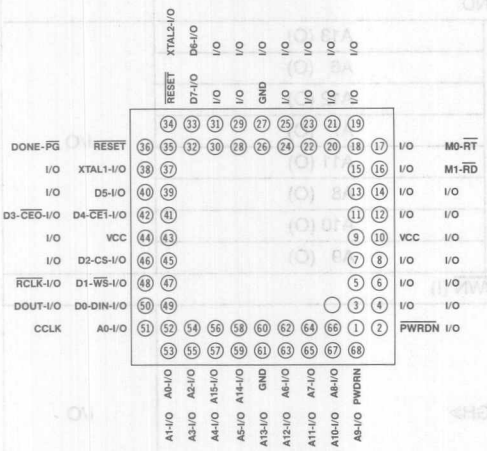
SYMBOL	DESCRIPTION		-20		-33		UNIT
			MIN	MAX	MIN	MAX	
t_{DRT} ①	RTRIG	PROG setup	300		300		ns
t_{RTH} ②		RTRIG HIGH	250		250		ns
t_{RTCC} ③	CCLK	RTRIG setup	100		100		ns
t_{CCRD} ④		RDATA delay		100		100	ns

Notes: 1. CCLK and DOUT timing are the same as for slave mode.
2. DONE/PROG output/input must be HIGH (device programmed) prior to a positive transition of RTRIG (M0).

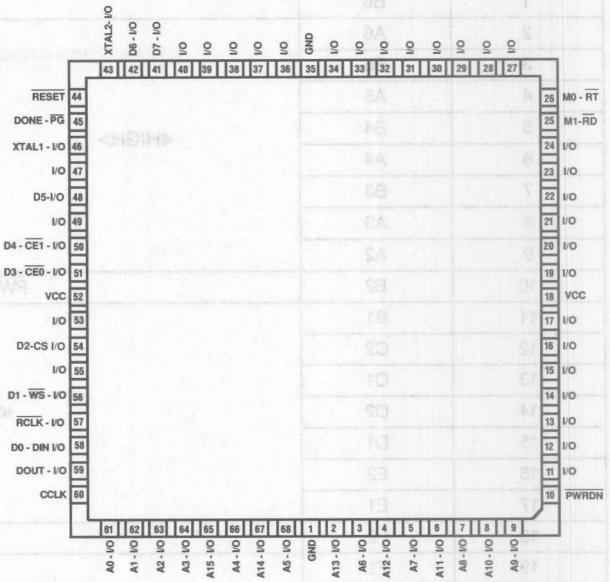


Device Pinouts

M2064
Logic Cell Array
68 Pin Grid Array
Top View

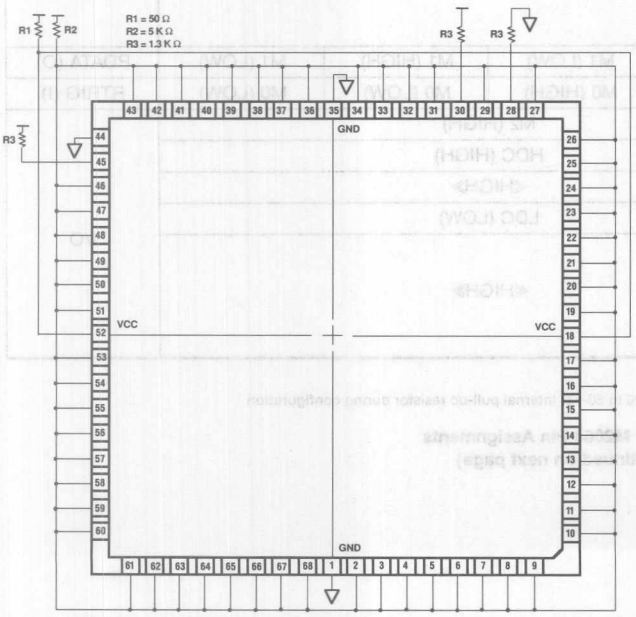


M2064
Logic Cell Array
68 Pin
Leadless Chip Carrier



Burn-In Circuitry

Condition C
Static Burn-In



Condition D
Dynamic Burn-In
Contact factory

Configuration Pin Assignments

68-PIN LCC	68-PIN PGA	CONFIGURATION MODE: ≪M2: M1: M0≫				USER OPERATION
		SLAVE <1:1:1>	PERIPHERAL <1:0:1>	MASTER-HIGH <1:1:0>	MASTER-LOW <1:0:0>	
1	B6	GND				
2	A6	≪HIGH≫		A13 (O)		I/O
3	B5			A6 (O)		
4	A5			A12 (O)		
5	B4			A7 (O)		
6	A4			A11 (O)		
7	B3			A8 (O)		
8	A3			A10 (O)		
9	A2			A9 (O)		
10	B2	$\overline{\text{PWRDWN}}$ (I)				
11	B1	≪HIGH≫				I/O
12	C2					
13	C1					
14	D2					
15	D1					
16	E2					
17	E1					
18	F2	VCC				
19	F1	≪HIGH≫				I/O
20	G2					
21	G1					
22	H2					
23	H1					
24	J2					
25	J1	M1 (HIGH)	M1 (LOW)	M1 (HIGH)	M1 (LOW)	RDATA (O)
26	K1	M0 (HIGH)	M0 (HIGH)	M0 (LOW)	M0 (LOW)	RTRIG (I)
27	K2	M2 (HIGH)				I/O
28	L2	HDC (HIGH)				
29	K3	≪HIGH≫				
30	L3	LDC (LOW)				
31	K4	≪HIGH≫				
32	L4					
33	K5					
34	L5					

≪HIGH≫ is high impedance with a 20 to 50-KΩ internal pull-up resistor during configuration

Table 1. M2064 Pin Assignments
(continued on next page)

Configuration Pin Assignments

68-PIN LCC	68-PIN PGA	CONFIGURATION MODE: ≪M2: M1: M0≫				USER OPERATION
		SLAVE <1:1:1>	PERIPHERAL <1:0:1>	MASTER-HIGH <1:1:0>	MASTER-LOW <1:0:0>	
35	K6	GND				
36	L6	≪HIGH≫				I/O
37	K7					
38	L7					
39	K8					
40	L8					
41	K9			D7 (I)		
42	L9			D6 (I)		
43	L10					XTL2 or I/O
44	K10	$\overline{\text{RESET}}$ (I)				
45	K11	DONE (O)				PROG (I)
46	J10	≪HIGH≫				XTL1 or I/O
47	J11					I/O
48	H10			D5 (I)		
49	H11					
50	G10		$\overline{\text{CS0}}$ (I)	D4 (I)		
51	G11		$\overline{\text{CS1}}$ (I)	D3 (I)		
52	F10	VCC				
53	F11	≪HIGH≫				I/O
54	E10		$\overline{\text{CS2}}$ (I)	D2 (I)		
55	E11					
56	D10		$\overline{\text{WRT}}$ (I)	D1 (I)		
57	D11			$\overline{\text{RCLK}}$		
58	C10	DIN (I)		D0 (I)		
59	C11	DOUT (O)				
60	B11	CCLK (I)	CCLK (O)			CCLK (I)
61	B10	≪HIGH≫		A0 (O)		I/O
62	A10			A1 (O)		
63	B9			A2 (O)		
64	A9			A3 (O)		
65	B8			A15 (O)		
66	A8			A4 (O)		
67	B7			A14 (O)		
68	A7			A5 (O)		

≪HIGH≫ is high impedance with a 20 to 50-KΩ internal pull-up resistor during configuration

Table 1. M2064 Pin Assignments

3

Pin Description

$\overline{\text{PWRDWN}}$
An active low power-down input stops all internal activity to minimize VCC power and puts all output buffers in a high-impedance state. Configuration is retained, however, internal storage elements are Reset. When the $\overline{\text{PWRDWN}}$ pin returns HIGH, the device returns to operation with the same sequence of reset, buffer enable and DONE, $\overline{\text{PROGRAM}}$ as at the completion of configuration.

M0, RTRIG
As Mode 0, this input and M1, M2 are sampled before the start of configuration to establish the configuration mode to be used.

As a read trigger, an input transition to a HIGH, after configuration is complete, will initiate a readback of configuration and storage element data. This operation may be limited to a single request, or be inhibited altogether, by selecting the appropriate readback option when generating the bit stream.

M1, $\overline{\text{RDATA}}$
As Mode 1, this input and M0, M2 are sampled before the start of configuration to establish the configuration mode to be used.

As an active-low read data; after configuration is complete, this pin is the output of the readback data.

M2
As Mode 2, this input and M0, M1 are sampled before the start of configuration to establish the configuration, mode to be used. After configuration, this pin becomes a user-programmable I/O.

HDC
High during configuration is held at a HIGH level by the LCA until after configuration. It is intended to be available as a control indication that configuration is not complete. After configuration, this pin is a user I/O.

$\overline{\text{LDC}}$
Low during configuration is held at a LOW level by the LCA until after configuration. It is intended to be available as a control indication that configuration is not completed. It is particularly useful in master mode as a LOW enable for an EPROM. After configuration, this pin is a user I/O. If used as a LOW EPROM enable, it should be programmed as a HIGH after configuration.

$\overline{\text{RESET}}$
This is an active-low input which has three functions. Prior to the start of configuration, a LOW input will delay the start of the configuration process. An internal circuit senses the application of power and begins a minimal time-out cycle on the order of 100 ms. When the time-out and $\overline{\text{RESET}}$ are complete, the levels of the "M" mode lines are sampled and configuration begins. If $\overline{\text{RESET}}$ is asserted during a configuration, the LCA is reinitialized and will restart the configuration at the termination of RESET. If $\overline{\text{RESET}}$ is asserted after configuration is complete, it will provide an asynchronous reset of all IOB and CLB storage elements of the LCA.

DONE, $\overline{\text{PROG}}$
The DONE open drain output is configurable with or without a pull-up resistor of about 3 KΩ. At the completion of configuration, the circuitry of the LCA becomes active in a synchronous order and one configuration clock cycle later DONE is asserted. Once configuration is done, a HIGH-to-LOW transition of this program pin will cause an initialization of the LCA and start a reconfiguration if that mode is selected in the current configuration.

XTL1
This user I/O pin may be configured to operate as the output of an amplifier usable with an external crystal and bias circuitry to form an oscillator.

XTL2
This user I/O pin may be configured to operate as the input of an amplifier usable with an external crystal and bias circuitry to form an oscillator.

CCLK
During configuration, configuration clock is an output of an LCA in either master or peripheral mode. LCAs in slave mode use it as a clock input. During a readback operation, it is an input clock for the configuration data being output.

DOUT
This user I/O pin is used during configuration to output serial configuration data out for daisy-chained slaves' data in.

DIN
This user I/O pin is used as serial data in during slave or peripheral configuration. This pin is D0 in master configuration mode.

$\overline{\text{CSO}}$, $\overline{\text{CS1}}$, CS2, $\overline{\text{WRT}}$
These four inputs represent a set of signals, three active low and one active high, which are used in the peripheral mode to control configuration data entry. The assertion of all four generates a LOW CCLK and shifts DOUT data. The removal of any assertion clocks in the DIN data present and causes a HIGH CCLK. In master mode, these pins become part of the parallel configuration byte (D4, D3, D2, D1). After configuration is complete, they are user-programmed I/O.

$\overline{\text{RCLK}}$
During master mode configuration, this pin represents a read clock of an external memory device. After configuration is complete, this pin becomes a user-programmed I/O.

D0-D7
This set of eight pins represents the parallel configuration data byte for the master mode. After configuration is complete, they are user-programmed I/O.

A0-A15
This set of sixteen pins presents an address output for an external configuration memory during master mode. After configuration is complete, they are user-programmed I/O. A12 through A15 are not available in packages with less than sixty-eight pins.

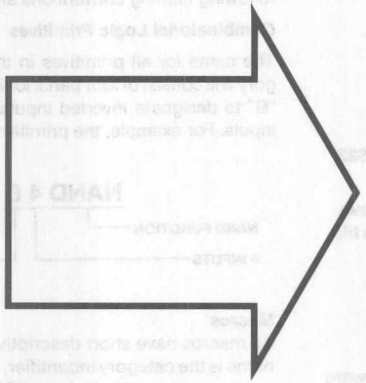
I/O
A pin which may be programmed by the user to be input and/or output following configuration. Some of these pins present a high-impedance pull-up or perform other functions before configuration is complete.

REVISIONS

REV No.	DATE	DESCRIPTION	DATE CODE OF EFFECT
IE8	8/87	Initial Release	8750

Monolithic Memories, Inc. Code Identification No. 50364	Data Sheet No. 10009

REVISIONS			
REV No.	DATE	DESCRIPTION	DATE CODE OF EXPECT
IEE	8/97	Initial Release	8750



NEW PRODUCT BRIEF

Daisy Schematic Entry Interface LCA-MDS33

Features

- **Design entry to XACT™ via the Daisy DEDII and ACE schematic editors.**
- **Macro library of 113 standard logic family equivalents derived from the XACT Macro Library**
- **Library of logic symbols including all two-input, three-input, and four-input AND, OR, and XOR gates plus storage, input/output, and clock elements**
- **User control for flagging critical paths for the LCA-MDS23 Automatic Placement and Routing Program**
- **Automatic partitioning and conversion of schematic drawings to a Monolithic Memories' Logic Cell™ Array design file**
- **Output compatibility with XACT Design Editor and the Automatic Placement and Routing Program**

Description

The LCA-MDS33 Daisy Schematic Entry Interface is a software package developed specifically to enable the users of Daisy workstations to create Logic Cell Array (LCA) designs using the Daisy DEDII or ACE schematic editors. The package includes an LCA symbol library, a DRW2XNF netlist translator, and an XNF2LCA file format converter. The user creates the LCA design using the Daisy schematic editor and the LCA symbol library, then converts the design into an XACT-compatible netlist via the DRW2XNF netlist translator,then uses the XNF2LCA file format converter to create an XACT-readable design file.

LCA-MDS33 Daisy Schematic Entry Interface Macro Library

To create LCA designs using Daisy workstations, designers enter schematics using either DEDII or ACE with the Monolithic Memories' Daisy LCA Macro Library. LCA schematics must contain only symbols from this library, or macros created with symbols from the library. User-created symbols should not be saved in the main library since future software updates may overwrite the library.

The Monolithic Memories' Daisy LCA Marcro Library contains 58 combinatorial logic primitives and 113 macros derived from the XACT Macro Library. Primitives represent the lowest level of logic symbols. Macros are logic functions implemented by the use of primitives and/or other macros. The library consists of the primitives and eight categories of macros.

Macro Naming Conventions

To allow easier access to the available macros in the library, the following naming conventions are used:

Combinatorial Logic Primitives

The name for all primitives in the "combinatorial logic" category will consist of four parts: logic function, number of inputs, a "B" to designate inverted inputs, and the number of inverted inputs. For example, the primitive NAND4B2 represents:

Macros

All macros have short descriptive names. The first letter of the name is the category indentifier, e.g., all macros of the "general" category starts with the letter "G" and so on. However, there are some exceptions to these rules. Please refer to the LCA-MDS33 Macro Library listing for all macros.

For detailed information regarding LCA macros and parameter conventions, please refer to the LCA Macrocell Library manual.

XACT Development System

The XACT Development System provides a complete design system for specification, implementation and verification of designs using LCAs. Functional definitions of CLBs, IOBs, and interconnection are performed with a built-in, menu-driven interactive graphics editor. However, manual design entry using the XACT graphics editor can be replaced by one of the corresponding schematic editors and Schematic Entry Interface packages. Once the design is captured either by the XACT or alternative schematic capture package, it is converted into binary configuration data to be loaded into an LCA for configuration. It is important to note that the XACT Development System is needed to generate this configuration data for an LCA. Please refer to the LCA Development System Brochure for more information.

2175 Mission College Blvd. Santa Clara, CA 95054-1592 Tel: (408) 970-9700 TWX: 910-338-2376 TWX: 910-338-2374
Monolithic Memories MMI

LCA-MDS32 Macro Library

Combinatorial Logic Primitives: 58

ACLK	Alternate Clock Buffer
BCLK	Global Clock Buffer
BUF1	Buffer
INV1	Inverter
AND2	2-Input AND Gate
AND2B	2-Input AND Gate; All Inputs Inverted
AND2B1	2-Input AND Gate; 1 Input Inverted
AND3	3-Input AND Gate
AND3B	3-Input AND Gate; All Inputs Inverted
AND3B1	3-Input AND Gate; 1 Input Inverted
AND3B2	3-Input AND Gate; 2 Inputs Inverted
AND4	4-Input AND Gate
AND4B	4-Input AND Gate; All Inputs Inverted
AND4B1	4-Input AND Gate; 1 Input Inverted
AND4B2	4-Input AND Gate; 2 Inputs Inverted
AND4B3	4-Input AND Gate; 3 Inputs Inverted
NAND2	2-Input NAND Gate
NAND2B	2-Input NAND Gate; All Inputs Inverted
NAND2B1	2-Input NAND Gate; 1 Input Inverted
NAND3	3-Input NAND Gate
NAND3B	3-Input NAND Gate; All Inputs Inverted
NAND3B1	3-Input NAND Gate; 1 Input Inverted
NAND3B2	3-Input NAND Gate; 2 Inputs Inverted
NAND4	4-Input NAND Gate
NAND4B	4-Input NAND Gate; All Inputs Inverted
NAND4B1	4-Input NAND Gate; 1 Input Inverted
NAND4B2	4-Input NAND Gate; 2 Inputs Inverted
NAND4B3	4-Input NAND Gate; 3 Inputs Inverted
NOR2	2-Input NOR Gate
NOR2B	2-Input NOR Gate; All Inputs Inverted
NOR2B1	2-Input NOR Gate; 1 Input Inverted
NOR3	3-Input NOR Gate
NOR3B	3-Input NOR Gate; All Inputs Inverted
NOR3B1	3-Input NOR Gate; 1 Input Inverted
NOR3B2	3-Input NOR Gate; 2 Inputs Inverted
NOR4	4-Input NOR Gate
NOR4B	4-Input NOR Gate; All Inputs Inverted
NOR4B1	4-Input NOR Gate; 1 Input Inverted
NOR4B2	4-Input NOR Gate; 2 Inputs Inverted
NOR4B3	4-Input NOR Gate; 3 Inputs Inverted
OR2	2-Input OR Gate
OR2B	2-Input OR Gate; All Inputs Inverted
OR2B1	2-Input OR Gate; 1 Input Inverted
OR3	3-Input OR Gate
OR3B	3-Input OR Gate; All Inputs Inverted
OR3B1	3-Input OR Gate; 1 Input Inverted
OR3B2	3-Input OR Gate; 2 Inputs Inverted
OR4	4-Input OR Gate

OR4B	4-Input OR Gate; All Inputs Inverted
OR4B1	4-Input OR Gate; 1 Input Inverted
OR4B2	4-Input OR Gate; 2 Inputs Inverted
OR4B3	4-Input OR Gate; 3 Inputs Inverted
XOR2	2-Input Exclusive-OR Gate
XOR3	3-Input Exclusive-OR Gate
XOR4	4-Input Exclusive-OR Gate
XNOR2	2-Input Exclusive-NOR Gate
XNOR3	3-Input Exclusive-NOR Gate
XNOR4	4-Input Exclusive-NOR Gate

General: 8

GADD	Adder
GCOMP	Compare
GEQGT	Equal or Greater
GMAJ	Majority
GMUX	2-to-1 Mux
GPAR	Parity
GXOR	Exclusive-OR
GXTL	Crystal Oscillator (not supported in the BETA release)

Pads: 9

IOFF	IOB Configured as a Flip-Flop
PBUF	Buffered Input
PIN1	Input Pad
PINQ	Input Pad with Storage
PIO	Input/Output Pad
PIOQ	Input/Output Pad with Input Storage
POUT	Output Pad
POUTZ	Output Pad with 3-State Control
PREG	Output Pad with Input Storage

Latches: 7

LD	Data Latch
LDM	Data Latch with 2-Input Data Mux
LDMRD	Data Latch with 2-Input Data Mux, ResetDir
LDMSD	Data Latch with 2-Input Data Mux, SetDir
LDRD	Data Latch with ResetDir
LDSD	Data Latch with SetDir
LDSRD	Data Latch with SetDir, ResetDir

Flip-Flops: 33

FD	D Flip-Flop
FDC	D Flip-Flop with Clock Enable
FDCR	D Flip-Flop with Clock Enable, Reset
FDCS	D Flip-Flop with Clock Enable, Set
FDM	D Flip-Flop with 2-Input Data Mux
FDMR	D Flip-Flop with 2-Input Data Mux, Reset
FDMS	D Flip-Flop with 2-Input Data Mux, Set
FDMRD	D Flip-Flop with 2-Input Data Mux, ResetDir

FDMSD	D Flip-Flop with 2-Input Data Mux, SetDir
FDR	D Flip-Flop with Reset
FDRD	D Flip-Flop with ResetDir
FDS	D Flip-Flop with Set
FDSD	D Flip-Flop with SetDir
FDSRD	D Flip-Flop with SetDir, ResetDir
PDFF	D Flip-Flop with Positive Edge Clock
NDFF	D Flip-Flop with Negative Edge Clock
FRS	Set-Rest Flip-Flop with Reset Dominant
FSR	Set-Reset Flip-Flop with Set Dominant
FJK	J-K Flip-Flop
FJKRD	J-K (Set-Reset) Flip-Flop with ResetDir
FJKS	J-K Flip-Flop with Synchronous Set
FJKSD	J-K (Set-Reset) Flip-Flop with SetDir
FJKSRD	J-K (Set-Reset) Flip-Flop with SetDir, ResetDir
FT	Toggle Flip-Flop
FTR	Toggle Flip-Flop with Reset
FTRD	Toggle Flip-Flop with ResetDir
FTS	Toggle Flip-Flop with Set
FT0	Self Toggle Flip-Flop
FT0R	Self Toggle Flip-Flop with Reset
FT2	2-Input Toggle Flip-Flop
FT2R	2-Input Toggle Flip-Flop with Reset
FTP	Toggle Flip-Flop with ParEna
FTPRD	Toggle Flip-Flop with ParEna, ResetDir

Decoders: 7 total

D2-4	1-of-4 Decoder
D2-4E	1-of-4 Decoder with Enable
D3-8	1-of-8 Decoder
D3-8E	1-of-8 Decoder with Enable
74-42	1-of-10 Decoder with Low Output
74-138	1-of-8 Decoder with Enables, Low Output
74-139	1-of-4 Decoder with Enables, Low Output

Multiplexers: 9

M3-1	3-to-1 Mux
M3-1E	3-to-1 Mux with Enable
M4-1	4-to-1 Mux
M4-1E	4-to-1 Mux with Enable
M8-1	8-to-1 Mux
M8-1E	8-to-1 Mux with Enable
74-151	8-to-1 Mux with Enable, Complementary Outputs
74-152	8-to-1 Mux with Low Output
74-352	8-to-1 Mux with Low Output, Enables

Registers: 11

RD4	4-Bit Data Register
RD8	8-Bit Data Register
RD8CR	8-Bit Data Register with Clock Enable, Reset

RS8	8-Bit Shift Register
RS8CR	8-Bit Shift Register with Clock Enable,
RS8PR	8-Bit Shift Register with Parallel Enable, Reset
RS8R	8-Bit Shift Register with Reset
74-164	8-Bit Serial to Parallel Shift Register with ResetDir
74-194	4-Bit Bidirectional Shift Register with Clock Enable, ParEna, ResetDir
74-195	4-Bit Serial Shift Register with ParEna, Reset

Counters: 29

C2BCR	1-Bit Binary Counter with ClkEna, Reset
C2BCRD	1-Bit Binary Counter with ClkEna, ResetDir
C2BP	1-Bit Binary Counter with ParEna
C2BR	1-Bit Binary Counter with Reset
C2BRD	1-Bit Binary Counter with ResetDir
C4BCR	2-Bit Binary Counter with ClkEna, Reset
C4BCRD	2-Bit Binary Counter with ClkEna, ReserDir
C4BCP	2-Bit Binary Counter with ClkEna, ParEna
C4JCR	2-Bit Johnson Counter with ClkEna, Reset
C6JCR	3-Bit Johnson Counter with ClkEna, Reset
C8BCR	3-Bit Binary Counters with ClkEna, Reset
C8BCRD	3-Bit Binary Counters with ClkEna, ResetDir
C8BCP	3-Bit Binary Counters with ClkEna, ParEna
C8JCR	3-Bit Johnson Counter with ClkEna, Reset
C10BCRD	4-Bit BCD Counter with ClkEna, ResetDir
C10BCPRD	4-Bit BCD Counter with ClkEna, ParEna, ResetDir
C10BPRD	4-Bit BCD Counter with ParEna, Reset
C10JCR	5-Bit Johnson Counter with ClkEna, Reset
C12JCR	6-Bit Johnson Counter with ClkEna, Reset
C16BARD	4-Bit Binary Ripple Counter with ResetDir
C16BCPR	4-Bit Binary Counter with ClkEna, ParEna, Reset
C16BCPRD	4-Bit Binary Counter with ClkEna, ParEna, ResetDir
C16BCRD	4-Bit Binary Counter with ClkEna, ResetDir
C16BPRD	4-Bit Binary Counter with ParEna, ResetDir
C16BUDRD	4-Bit Binary Up-Down Counter with ParEna, ResetDir
C16JCR	8-Bit Johnson Counter with ClkEna, Reset
C256FCRD	8-Bit Modulo 256 Feedback Shift Register with ClkEna, ResetDir
74-160	4-Bit BCD Counter with ClkEna, ParEna, ResetDir
74-161	4-Bit Binary Counter with ResetDir

System Requirements and Limitations

The LCA-MDS33 Daisy Schematic Entry Interface provides the customer a way to utilize Daisy's schematic capture packages to enter an LCA design. However, some of the original Daisy workstations, such as Daisy Logicians™, are not capable of supporting the XACT Development Software, the key application in creating the LCA configuration data. In such cases, the data generated by the Daisy Schematic Entry Interface must be physically transported to a hardware platform which can support XACT. Depending on the workstation, the level of software compatibility differs widely.

Hardware Configuration

SOFTWARE	DAISY LOGICIAN V,D	DAISY PERSONAL LOGICIAN™ with CMG**	DAISY PERSONAL LOGICIAN with CMG and BMG***	DAISY* ENTRY SYSTEM with EGA
Schematic Capture - DEDII	X	X	X	X
Schematic Capture - ACE			X	X
DANCE / ADANCE	X	X	X	X
DRINK	X	X	X	X
LCA Netlist Extract - DRW2XNF	X	X	X	X
DOS File Forma' Conversion - N	X	X	X	X
DOS File Format Conversion - DOSCOPY		X	X	X
Daisy Local Area Network - Daisy Ethernet	X	X	X	X
LCA File Format Conversion - XNF2LCA			X	X
LCA Design Editor - XACT V1.3			X	X

* IBM™ PC-AT™ based system.
** CMG = Standard Daisy Character Map Graphics Card.
*** BMG = Personal Logician Bit Map Graphics Card.

4

Valid Hardware Combinations and Design Implementation Steps

I. **Logician Based System 1**
Hardware Required: Daisy Logician V or D
Daisy Local Area Network
Daisy Personal Logician with CMG
IBM PC-AT/XT or Fully Compatible

1. On a Logician V or D with V5.02 DNIX:
   DEDII : Schematic Capture System
   DANCE : Daisy Connectivity Extractor
   DRINK : Daisy Resolving Linker
   DRW2XNF : LCA Netlist Extractor
2. Using the Daisy LAN, transfer design data files to a Personal Logician (PC-AT based).
3. Convert the design data files to DOS format and create 5-1/4 inch diskettes.
4. Transfer the design files via 5-1/4 inch floppies to an IBM PC-AT/XT or fully compatible machine capable of supporting XACT.
5. Finish LCA design using XACT.

II. **Logician Based System 2**
Hardware Required: Daisy Logician V or D
Daisy Local Area Network
Daisy Personal Logician with CMG and BMG

1. On a Logician V or D with V5.02 DNIX:
   DEDII : Schematic Capture System
   DANCE : Daisy Connectivity Extractor
   DRINK : Daisy Resolving Linker
   DRW2XNF: LCA Netlist Extractor
2. Using the Daisy LAN, transfer design data files to a Personal Logician (PC-AT based).
3. Convert the design data files to DOS format.
4. Invoke XACT under a DOS environment.
5. Finish LCA design using XACT.

III. **Personal Logician Based System 1**
Hardware Required: Daisy Personal Logician with CMG
IBM PC-AT/XT or Fully Compatible

1. On a Personal Logician with V5.02 DNIX:
   DEDII : Schematic Capture System
   DANCE : Daisy Connectivity Extractor
   DRINK : Daisy Resolving Linker
   DRW2XNF : LCA Netlist Extractor
2. Convert the design data files to DOS format.
3. Transfer design to an IBM PC-AT/XT via 5-1/4 inch floppies.
4. Finish LCA design using XACT.

IV. **Personal Logician Based System 2**
Hardware Required: Daisy Personal Logician with CMG and BMG

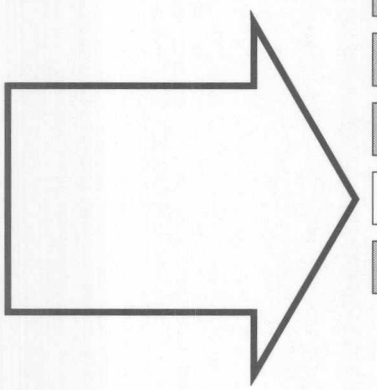
1. On a Personal Logician with V5.02 DNIX:
   DEDII or ACE : Schematic Capture System
   DANCE : Daisy Connectivity Extractor
   DRINK : Daisy Resolving Linker
   DRW2XNF : LCA Netlist Extractor
2. Convert the design data files to DOS format.
3. Invoke XACT under a DOS environment.
4. Finish LCA design using XACT.

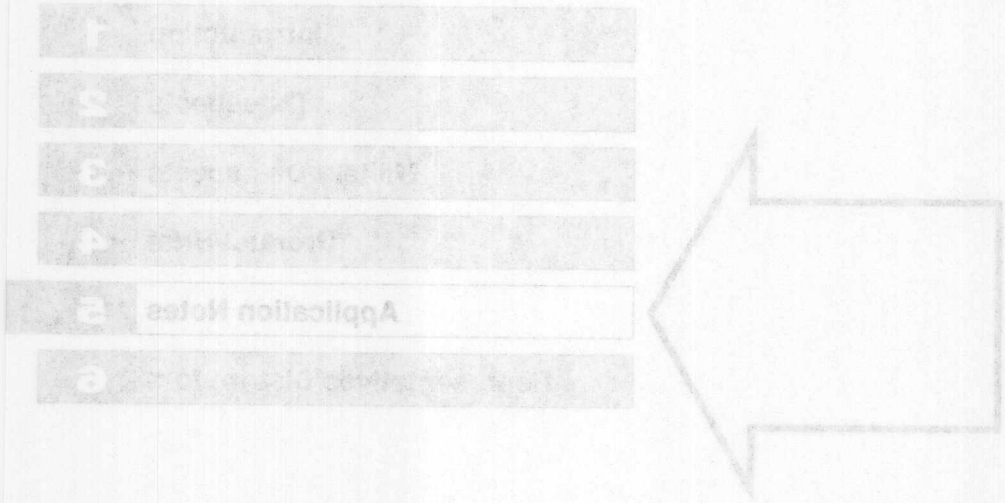
V. **Daisy Entry System**
Hardware Required: Entry System (IBM PC-AT with BMG and EGA)

1. On the Entry System with V5.02 DNIX:
   DEDII or ACE : Schematic Capture System
   DANCE : Daisy Connectivity Extractor
   DRINK : Daisy Resolving Linker
   DRW2XNF : LCA Netlist Extractor
2. Convert the design data files to DOS format.
3. Invoke XACT under a DOS environment.
4. Finish LCA design using XACT.

Related Products Ordering Information

PRODUCT	INFORMATION
LCA-MDS21	XACT Development System
LCA-MDS22	P-SILOS™ Simulator
LCA-MDS23	Automatic Placement and Routing Program
LCA-MDS24-48N	XACTOR™ In-Circuit Emulator for 48-Pin DIP (Includes one LCA-MDS26 and one LCA-MDS27-48N)
LCA-MDS24-68NL	XACTOR In-Circuit Emulator for 69-Pin PLCC (Includes one LCA-MDS26 and one LCA-MDS27-68NL)
LCA-MDS24-68P	XACTOR In-Circuit Emulator for 69-Pin PGA (Includes one LCA-MDS26 and one LCA-MDS27-68P)
LCA-MDS24-84NL	XACTOR In-Circuit Emulator for 84-Pin PLCC (Includes one LCA-MDS26 and one LCA-MDS27-84NL)
LCA-MDS24-68P	XACTOR In-Circuit Emulator for 84-Pin PGA (Includes one LCA-MDS26 and one LCA-MDS27-84P)
LCA-MDS26	Universal Emulation Pod
LCA-MDS27-48N	Emulation Header Cable for 48-Pin DIP
LCA-MDS27-68NL	Emulation Header Cable for 68-Pin PLCC
LCA-MDS27-68P	Emulation Header Cable for 68-Pin PGA
LCA-MDS27-84NL	Emulation Header Cable for 84-Pin PLCC
LCA-MDS27-84P	Emulation Header Cable for 84-Pin PGA
LCA-MSC21	XACT Development System Annual Support Agreement





Configuring the LCA Device

By Ed Valleau

5

2175 Mission College Blvd. Santa Clara, CA 95054-1592 Tel: (408) 970-9700 TWX: 910-338-2374
TWX: 910-338-2376
Monolithic Memories MMI

Configuring the LCA Device

By Ed Valleau

Introduction

The Logic Cell™ Array device from Monolithic Memories is a new revolution in programmable logic. It offers the user gate array logic density while still providing the versatility of a programmable logic device (PLD). Its architecture is based on an array of static RAM cells, which must be configured when power is initially applied to the circuit. This gives the LCA device a high degree of flexibility since the functional logic contained in the device may be changed during development, thus shortening prototyping cycles. To update existing equipment, logic circuits may be modified with new configuration data. It is also possible to reconfigure the LCA device "on the fly" without the need for hardware changes. This would prove specifically useful in systems that need to change their protocols of operation dynamically.

Configuration data is stored in a nonvolatile source and read by the LCA device at "power up". The time it takes to perform a configuration is variable, from 12 to 34 ms depending on device type and mode of configuration. A typical storage medium for the LCA's configuration data would be an EPROM or EEPROM, which would be connected to the the LCA and accessed immediately after "power up". When the configuration process is complete and the device is fully functional, the EPROM or EEPROM may be put into a three-state condition and left deselected until the next configuration cycle.

There are a number of different modes of LCA device configuration, each mode being better suited to a particular operating environment or application than another mode. The following information outlines these various modes.

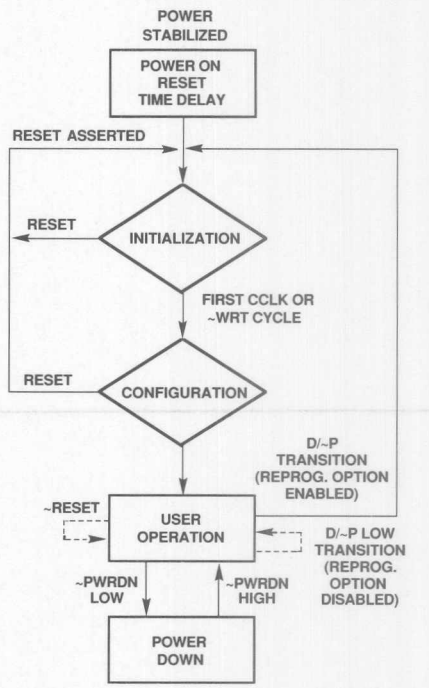


Figure 1. Configuration Sequence

Overview of the Configuration Process

The LCA device is comprised of three distinct programmable elements: Configurable Logic Blocks (CLBs), Input/Output Blocks (IOBs) and Programmable Interconnect. Because the device architecture is based on static RAM cells, the configuration of the CLBs, IOBs and Programmable Interconnect is random and indeterminate when power is first applied to the device. The LCA device goes through two states prior to being functionally operational. The first is a general initialization state, followed by a configuration sequence. (See Figure 1.)

The device uses a number of dedicated input/output pins to control the loading process. Configuration sequentially programs the RAM cells, ultimately creating functionally operational and interconnected logic blocks. The entire procedure is comparable to programming a PLD except that a conventional PLD is nonvolatile, and maintains its logic functionality when power is removed from the circuit.

The smaller of the two LCA devices, the M2064, is a 64-CLB device which requires exactly 12,040 binary bits of information to complete the configuration process. The M2018 is a 100-CLB device which needs 17,880 bits of information.

Table 1 shows the different modes that can be selected via the three dedicated mode input pins M0, M1 and M2. These will usually be hardwired to select a single mode. The LCA device reads the digital code applied to the mode pins prior to configuration, then enters one of five specific modes before becoming a functional logic system. The exact number of bits for successful configuration must be read into the LCA device, partial configuration is not possible.

CONFIGURATION MODE	SLAVE MODE	PERIPHERAL MODE	MASTER-HIGH MODE MASTER-LOW MODE	MASTER SERIAL MODE
Mode Selection code (M0:M1:M2)	1:1:1	1:0:1	0:0:1 (Master-Low) 0:1:1 (Master-High	0:0:0
Configuration data	Bit-serial	Bit-serial	Byte-parallel	Bit-serial
Automatic loading	No	No	Yes	Yes
Programming source	User Logic or Another LCA (Note 2)	CPU Data Bus Memory	External Byte-wide Memory	External Serial Memory
Number of user I/O pins required	2	6	25	3
Configuration time	Source Dependent (Note 1)	Source Dependent (Note 1)	12-24 ms (M2064) 17-34 ms (M2018) (Note 3)	12-24 ms (M2064) 17-34 ms (M2018) (Note 3)

Notes:
1. The minimum time in any case is approximately 12 ms for the M2064 and 17 ms for the M2018.
2. Also used by Monolithic Memories' XACTOR for In-Circuit Emulation.
3. This parameter depends on internal timing circuits and is manufacturing process-dependent. Therefore it may vary from device to device within the limits shown.

Table 1. Comparison of Configuration Modes

Choice of Configuration Mode

The choice of a configuration mode is influenced by the actual operating environment. For example, questions that might arise are: is the LCA being used as a standalone logic unit, or with a microprocessor? Can it be configured by a serial link? Other considerations include whether pins used for configuration are also used for functional operation, and whether or not these pins should be isolated from activating external logic during configuration. The designer has a choice of configuration modes for multiple LCA designs, and can use either parallel or serial support for a master mode LCA device which in turn can configure a slave or number of slave LCA devices.

The Configuration Pins

The pins used to configure the LCA device come in two forms: dedicated configuration pins which are used exclusively for configuring the LCA device, and multifunction configuration pins which can be used as general-purpose I/Os after configuration has been completed.

The Six Dedicated Configuration Pins

PIN	DESCRIPTION
M0,M1	Mode Select Pins
CCLK	Configuration Clock
~Reset	Master Reset
D/~P	Done/~Program
~PWRDN	Power Down

The Multi-Function Configuration Pins

PIN	DESCRIPTION	USAGE
M2	Mode Select	Present in all configuration modes
Din	Configuration Data In	
Dout	Configuration Data Out	
HDC	High During Configuration	
LDC	Low During Configuration	
~CS0,~CS1,CS2	Chip Selects	Peripheral Mode Only
~Wrt	Write Strobe	
~RCLK	Read Strobe	
A0-A15	Address Lines	Master Mode Only
D0-D7	Input Data Lines	

Pin Names and Functional Description

Done/Program Pin D/~P

The Done/~Program D/~P pin performs a dual function. It is an open-drain output with an internal programmable pull-up resistor. During configuration the pin is an output that is driven LOW by the LCA and can be monitored by external logic to determine whether or not the LCA is ready for functional use. When configuration is completed this pin is pulled HIGH by an internal programmable pull-up resistor of 3KΩ value. In a multi-LCA environment, the D/~P pin goes HIGH one clock cycle before the configuration is complete allowing time for user I/O signals to propagate between other LCA devices before entering the functional mode of operation. The LCA may be reconfigured at any time by pulling the D/~P input LOW with an external logic open-drain (open-collector) driver. When the internal logic recognizes a LOW input it drives the D/~P output LOW forcing a reconfiguration cycle.

There is a time delay of several microseconds before an active LOW input is recognized, so random triggering due to short noise bursts is highly unlikely. If the D/~P input is prevented from going HIGH immediately after configuration, then further reconfigurations will be suppressed and functional operation will not commence until that input is permitted to go HIGH. In multiple LCA designs the D/~P pins are wire-OR 'ed, so the last LCA device to be configured will prevent other, already configured devices, from functioning. When the last device is configured the composite wire-OR line goes HIGH and all LCA devices enter a functional mode simultaneously.

~RESET

The ~RESET input is active LOW and is used to start the initialization process before or during configuration, but not during functional operation. Figure 1 shows the initialization and configuration procedure as a flow diagram. Asserting the ~RESET line will put the LCA device into an initialization mode, aborting the current cycle of configuration.

Once configuration is complete the ~RESET input takes on a different role. It may be used to reset the device as if the device were a functional logic system. When asserted, internal registers and/or latches are reset. The reset condition may then be relaxed and functional operation resumed without the generation of a reconfiguration cycle.

To initiate reconfiguration the D/~P pin should be used instead of the ~RESET input. When ~RESET is LOW at the initialization cycle, its LOW-to-HIGH transition will sample the logic condition at the mode input pins. The logic state sampled determines the mode of configuration. Once configuration has started these pins are not required to be held. However, it is recommended that these pins remain unchanged until the configuration phase is completed.

MODE CONFIGURATION INPUTS M0, M1, and M2

M0, M1 and M2 are the Configuration Mode Select Pins. They force the LCA device into the selected configuration mode on the rising edge of ~RESET. Pins M0 and M1 are dedicated mode inputs having no general-purpose I/O capability. They will usually be hardwired to logic HIGH or LOW conditions. For all modes except the master serial mode, M2 is HIGH and is pulled HIGH internally during configuration. After configuration this pin can be used for general I/O functions. M2 and M0 both have internal pull-up resistors. M1 should be tied to VCC or GND. Table 2 lists five of the modes of operation. The only mode that requires a logic LOW on the input to M2 is the Master Serial Mode which is used with a dedicated serial port EPROM.

M0	M1	M2	MODE SELECT
0	0	0	Master Serial Mode
0	0	1	Master LOW Mode
0	1	1	Master HIGH Mode
1	0	1	Peripheral Mode
1	1	1	Slave Mode

Table 2.

PWRDWN~

The PWRDWN~ input is an active LOW power-down pin that can be used to reduce power consumption of the LCA device when it is not being used. The LCA device is then considered off-line because all I/O lines are disabled while internal configuration is maintained. The VCC input may be reduced to 2.0 Volts and configuration data will not be lost. With internal logic disabled and I/O pins in a high impedance condition, the D/~P pin is forced LOW and all internal storage elements become cleared. It is essential that this pin is used only while D/~P is HIGH after initialization and configuration. It is common to tie this input to a valid logic HIGH if the power-down feature is not required.

Non-Dedicated Pins used in all Configuration Modes

DIN and DOUT

DIN (configuration Data IN) is used as a serial data port into the LCA device. Individual data bits are applied to this input and clocked into the device by rising clock edges applied to the CCLK input. DIN is used in both slave and peripheral modes.

DOUT (configuration Data OUT) is used to configure multiple LCA devices in a daisy chain. The DOUT signal from the preceding LCA device drives the DIN of the succeeding device and is synchronized to rising edge of the clock pulse on the CCLK line.

CCLK

CCLK (Configuration CLocK) as described above is used to synchronize the serial data stream into the data input DIN pin. The CCLK can be an input pin or output pin depending on the mode of configuration. It is an output for every mode of configuration except for the slave mode. The CCLK reverts to an input in the slave mode, and can be driven either by an external source or the CCLK output of another LCA device configured in the master mode. In the operational mode the CCLK is an input to enable configuration data to be read back from the LCA via the M0 and M1 pins, which have the dual function of Readback Trigger (RT) and Readback Data (RD), respectively. CCLK has an internal pull-up resistor which allows the input to be pulled HIGH when not in use.

RCLK

The Read CLocK is used as an output signal which goes LOW when the LCA device expects to see valid input data, and HIGH when the address bus contents are changing states. It can be used to enable the external EPROM, and in the master serial mode can be used to clock an external address counter. This pin is used to clock the CLK input of the serial EPROM, a support device for the LCA.

HDC and LDC

High During Configuration (HDC) and Low During Configuration (LDC) are driven HIGH and LOW, respectively, for the duration of the configuration process. They are used to control external logic during configuration. When configuration has been completed HDC and LDC become general-purpose input/output pins. All of the other I/O pins not involved in the actual configuration process are connected to internal pull-ups to VCC, which are removed after configuration.

CHIP SELECT INPUTS ~CS0 ~CS1 CS2 ~WRT

~CS0, ~CS1, and CS2 are the chip select pins used during the Peripheral Configuration Mode only. The three enables can be used to map the LCA device to a specific address. Active LOW inputs to ~CS1 and ~CS0, and an active HIGH input to CS2 will select the LCA device as if it were a memory or peripheral location mapped into the address space of a microprocessor system.

~WRT in the Peripheral Configuration Mode is the same as CCLK in the slave mode. A single data bit is transferred from the data bus into the LCA device on each rising edge of ~WRT.

A0-A15 and D0-D7

The address and data pins are used in the master parallel modes only, and are converted to general-purpose input/output pins after successful configuration. A0-A15 are the address lines used to access the external EPROM. In the Master Low mode, address location 0000 hexadecimal is read first, and the addresses increment each time a data byte is read until all of the configuration data is loaded into the LCA device. The D/~P then goes HIGH indicating that the device is loaded. The address counter outputs convert to general-purpose I/O, unless the D/~P pin is held LOW by the wire-OR function of a D/~P output from a slave device. In that case the counters will continue to count, downloading the slave's configuration data. Counting will continue for more than one slave. In the Master High Mode, address FFFF hexadecimal is accessed first, and the address lines are decremented. This allows the LCA device to share addressing space with an EPROM or EEPROM used by the system. D0-D7 are the data lines connected to the external memory device.

Slave Configuration <M0 M1 M2> = <1 1 1>

The Slave Configuration Mode (Figure 2a) is simple to implement requiring only three pins, two lines to perform handshake and synchronization and one line for data transmission. Slave configuration is used in the XACT Development System to download data to the support hardware. In this mode data is presented to the LCA device as a serial bit stream, which is transferred to the device as if it were a very large shift register. The data is presented to the DIN pin and sequentially clocked into the device using the CCLK input. When the device is completely loaded, the D/~P pin goes HIGH to confirm that configuration is complete and that the device is ready to function as a programmed unit.

In Figure 2a the LCA device is shown connected to a microcomputer/microprocessor port with the data line (D0) connected to the DIN input. The rising edge of the strobe output will clock valid data on D0 into the LCA. The process continues until the D/~P confirms that configuration is complete to the microcomputer via the data input (D7). The microcomputer can poll D7 to

PIN NAME	APPLICABLE CONFIGURATION MODE(S)					FUNCTION DURING CONFIGURATION	FUNCTION DURING USER OPERATION
	S	P	MH	ML	MS		
M0	•	•	•	•	•	Mode select 0 (I)	Readback trigger (I)
M1	•	•	•	•	•	Mode select 1 (I)	Readback data out (O)
M2	•	•	•	•	•	Mode select 2 (I)	<User I/O>
D/P (Note 1)	•	•	•	•	•	Indicates when configuration process is done (O)	Initiates/Inhibits Reconfiguration (I)
RESET (Note 1)	•	•	•	•	•	Abort/Restart configuration (I)	Master clear of all internal Flip-Flops (I)
CCLK	•	•	•	•	—	Configuration clock (See Notes 1 and 2)	Readback clock (I)
DIN	•	•	—	—	•	Configuration data in (I)	<User I/O> (Note 3)
DOUT	•	•	•	•	•	Configuration data out (O)	<User I/O>
HDC	•	•	•	•	•	Logic HIGH (O)	<User I/O>
LDC	•	•	•	•	•	Logic LOW (O)	<User I/O>
A0-A15	—	—	•	•	—	Address bus (O)	<User I/O>
D0-D7	—	—	•	•	—	Data bus (I)	<User I/O> (Note 3)
RCLK	—	—	•	•	•	Read clock (O)	<User I/O>
WRT	—	•	—	—	—	Write strobe (I)	<User I/O>
CS0	—	•	—	—	—	Chip select 0 (I)	<User I/O>
CS1	—	•	—	—	—	Chip select 1 (I)	<User I/O>
CS2	—	•	—	—	—	Chip select 2 (I)	<User I/O>

Notes:
1. The $\overline{\text{RESET}}$, CCLK, and D/$\overline{\text{P}}$ pins have multiple functions. See text for further details.
2. During Slave mode configuration, the CCLK pin is an input, while for all other modes, it is an output.
3. DIN and D0 are the same physical pins but are associated with different configuration modes.

Abbreviations:
S = Slave
P = Peripheral
I = Input
O = Output
MH = Master high
ML = Master low
MS = Master serial

Table 3. Summary of Pins Used for Configuration

determine the configuration status, or set input D7 to generate an interrupt. Another application for the slave mode would be in a multiple LCA design where one LCA device can serially load subsequent slave LCA devices arranged in a daisy chain. Using LCA devices connected as master and slave or slaves is discussed more fully in a later section.

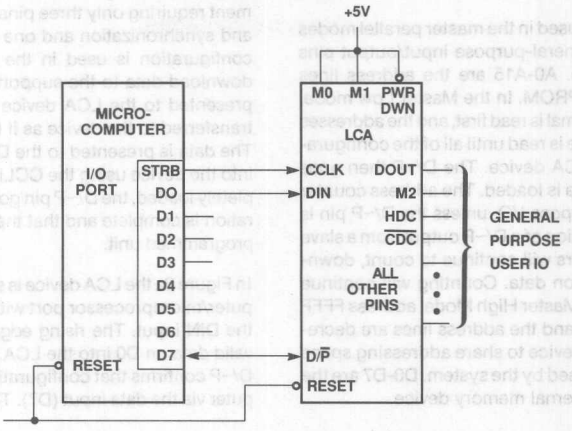


Figure 2a. Slave Mode

PIN NAME	PIN NUMBER		PIN TYPE	VALUE DURING CONFIGURATION	DESCRIPTION
	PLCC	DIP			
Fixed, Non-programmable Pins					
M0	26	18	Input	HIGH	Mode Select
M1	25	17	Input	HIGH	Mode Select
CCLK	60	42	Input	<Clock>	Configuration Clock
RESET	44	31	Input	HIGH	Master Reset
D/$\overline{P}$	45	32	Output	LOW	Done/Program
PWRDWN	10	7	Input	HIGH	Power-down
User-Programmable Pins					
M2	27	19	Input	HIGH	Mode Select
DIN	58	40	Input	<Data>	Configuration Data In
DOUT	59	41	Output	<Data>	Configuration Data Out
HDC	28	20	Output	HIGH	Constant "1" Level
LDC	30	21	Output	LOW	Constant "0" Level

Figure 2b. Slave Mode Pin Summary

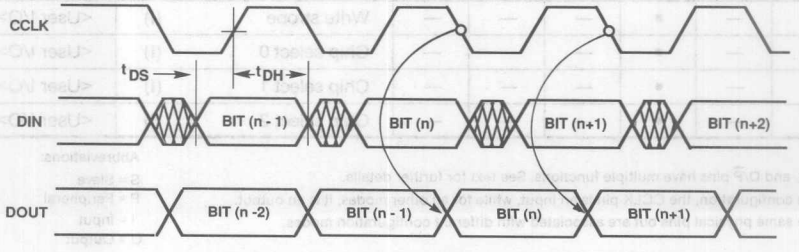


Figure 2c. Slave Mode Configuration Timing

Peripheral Mode Configuration <M0 M1 M2> = <1 0 1>

The Peripheral Mode (Figure 3a) is similar to the Slave Mode in that the data is presented in a bit serial form. In the Peripheral mode, however the LCA device is configured as a microprocessor-compatible peripheral device. The microprocessor can access the LCA device directly through its internal address bus, and map it through chip select logic. Control of the write operation is direct to the LCA device. Less hardware is required to connect the LCA to a microprocessor in this mode, and the loading of data is serial over a single data line. Synchronization is achieved by the falling edge of a ~WRT input, while data is set up on the DIN line and strobed by the falling edge of a microprocessor-generated write signal.

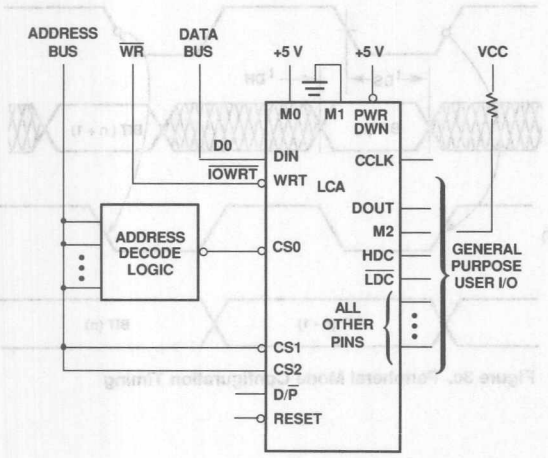


Figure 3a. Peripheral Mode

PIN NAME	PIN NUMBER		PIN TYPE	VALUE DURING CONFIGURATION	DESCRIPTION
	PLCC	DIP			
Fixed, Non-programmable Pins					
M0	26	18	Input	HIGH	Mode Select
M1	25	17	Input	LOW	Mode Select
CCLK	60	42	Output	<Clock>	Configuration Clock
$\overline{\text{RESET}}$	44	31	Input	HIGH	Master Reset
D/$\overline{\text{P}}$	45	32	Output	LOW	Done/Program
$\overline{\text{PWRDWN}}$	10	7	Input	HIGH	Power-down
User-Programmable Pins					
M2	27	19	Input	HIGH	Mode Select
DIN	58	40	Input	<Data>	Configuration Data In
DOUT	59	41	Output	<Data>	Configuration Data Out
$\overline{\text{CS0}}$	50	35	Input	LOW	Chip select (Active LOW)
$\overline{\text{CS1}}$	51	36	Input	LOW	Chip select (Active LOW)
CS2	54	37	Input	HIGH	Chip select
$\overline{\text{WRT}}$	56	38	Input	<Strobed>	Write enable (Active LOW)
HDC	28	20	Output	HIGH	Constant "1" Level
$\overline{\text{LDC}}$	30	21	Output	LOW	Constant "0" Level

Figure 3b. Peripheral Mode Pin Summary

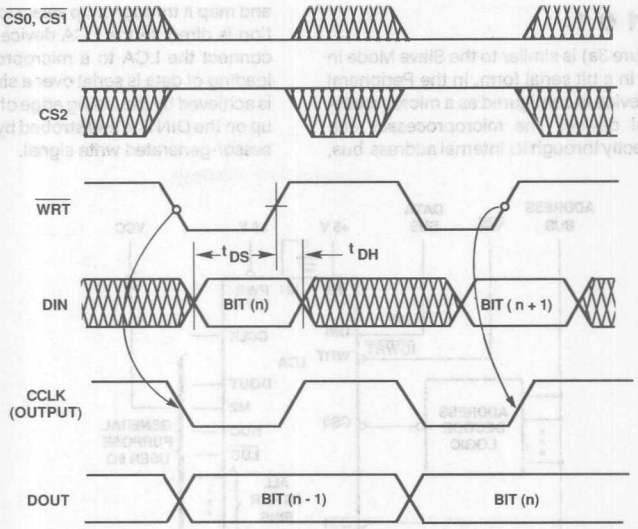


Figure 3c. Peripheral Mode Configuration Timing

Parallel Master Mode Configuration

MASTER MODE LOW <M0 M1 M2> = <0 0 1>

MASTER MODE HIGH <M0 M1 M2> = <0 1 1>

In the Master Low Parallel mode (Figure 4a) data is loaded in parallel at a rate determined by an on-chip timer in the LCA device. Configuration data is stored in an EPROM or EEPROM, and a parallel data path is connected from the memory output data lines D0-D7 to dedicated data inputs on the LCA device. There are sixteen address outputs, A0(LSB)-A15(MSB), connected to the EPROM. At the start of configuration in master mode low the LCA device sends an incrementing address starting at address 0000 hexadecimal and sequentially selects the configuration bytes. Data is serialized when loaded into the LCA device.

When configuration is complete the address and data lines become general-purpose I/O blocks. The D/~P pin is used to provide external logic, an indication that the LCA is currently functional or configuring.

The D/~P pin can be used to select the EPROM/EEPROM during configuration. An active LOW output applied to the memory's Chip Select input will select the device to read the configuration data and deselect it when configuration is complete. The method shown in Figure 4a is suggested for logic systems that have no host processor available to perform the configuration. A drawback is that address and data lines might require isolation from external logic circuits while the LCA device is configuring. The logic designer must determine which signals need isolation and which do not. Using IOBs which are not address and data lines might be a solution to avoid any conflict that might occur when external logic is driven during configuration. Also, the designer must avoid loading address and data lines with capacitive or inductive components. For example general IOBs should be used for circuits such as relaxation oscillators which require capacitive loads. Loading address or data lines with capacitors might lead to a violation of setup and hold times, causing erroneous configuration information to be read.

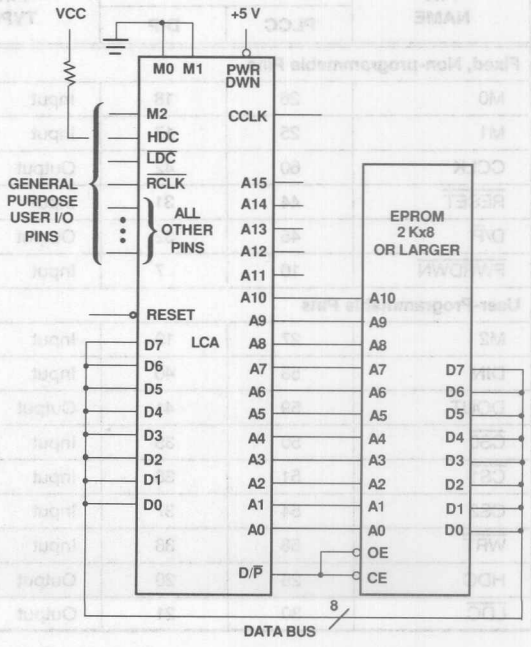


Figure 4a. Master Parallel Mode

In the Master Mode HIGH the initial starting address is hexadecimal FFFF, and the address decrements after each data byte is read. For both Master Mode HIGH and Master Mode LOW, the LCA device will set the D/~P pin HIGH after enough bits have been read to configure the one master LCA device. This permits configuration data to be stored in an EPROM that also holds microprocessor firmware. Only 1505 (5E1 hex) bytes are required for an M2064 LCA device and 2235 (8BB hex) bytes for the M2018, so large EPROMs can store configuration data for a number of LCA devices. Configuration information is concatenated in one EPROM and read by the master device, then sent to slave LCA devices as outlined in the section describing the configuring of multiple LCA devices.

Bytewide Master Mode Pin Summary

PIN NAME	PIN NUMBER PLCC	PIN NUMBER DIP	PIN TYPE	VALUE DURING CONFIGURATION	DESCRIPTION
Fixed, Non-programmable Pins					
M0	26	18	Input	LOW	Mode Select
M1	25	17	Input	LOW or HIGH	(Master-low mode) (Master-high mode)
CCLK	60	42	Output	<Clock>	Configuration Clock
$\overline{\text{RESET}}$	44	31	Input	HIGH	Master Reset
D/$\overline{\text{P}}$	45	32	Output	LOW	Done/Program
$\overline{\text{PWRDWN}}$	10	7	Input	HIGH	Power-down
User-Programmable Pins					
M2	27	19	Input	HIGH	Mode Select
DOUT	59	41	Output	<Data>	Configuration Data Out
HDC	28	20	Output	HIGH	Constant "1" Level
$\overline{\text{LDC}}$	30	21	Output	LOW	Constant "0" Level
$\overline{\text{RCLK}}$	57	39	Output	<Strobed>	Chip enable output
A0-Axx	—	—	Outputs	<Address>	Memory address bus
		48 DIP		A15 ... A11	A0
				3 5 6 4 2 1 48 47 46 45 44 43	
	68 PLCC			65 67 2 4 6 8 9 7 5 3 68 66 64 63 62 61	
D0-D7	—	—	Inputs	<Data>	Memory data bus
		48 DIP		D7	D0
				28 29 34 35 36 37 38 40	
	68 PLCC			41 42 48 50 51 54 56 58	

Table 4b. Master Mode Pin Summary

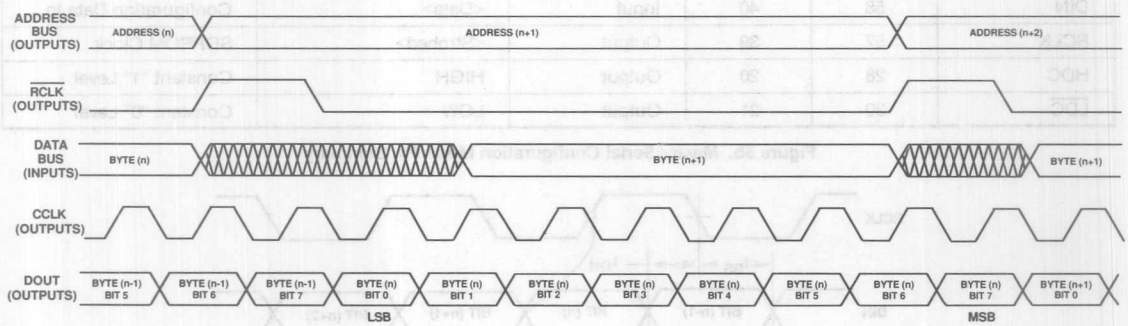


Figure 4c. Master Mode Configuration Timing

Serial Master Mode
<M0 M1 M2> = <0 0 0>

The Master Serial Mode (Figure 5) uses a Serial Data PROM (SDPROM) which has an internal address counter and a serial data port for the conversion of parallel data from the SDPROM to a single data output. Figure 5a shows the schematic diagram required for this mode. Configuration data is stored in the PROM section and downloaded to the LCA device in the most pin-efficient way, avoiding large numbers of address and data interconnections. The SDPROM is connected to the LCA device using only three pins. In this mode the LCA generates a clock signal which is used to increment the address counter built into the SDPROM. The SDPROM then sends data to the DIN pin on the LCA device. This mode is very similar to the Slave and Peripheral Modes in that data input is through the DIN pin. However, the SDPROM requires a clock input to increment its internal address counter which is supplied by the RCLK output of the LCA device. The configuration time is therefore determined by the LCA devices on chip counter. The D/~P input terminates the configuration process by disabling the SDPROM. One SDPROM can hold enough configuration data for three M2064 or two M2018 devices, and they can be cascaded for larger configuration arrays.

The data to be loaded into an LCA device is developed using the XACT Development Software, and is stored in one of two modes: a serial bit stream to be used in the Peripheral and Slave modes, or a 1500-byte PROM file for use in the Master Mode.

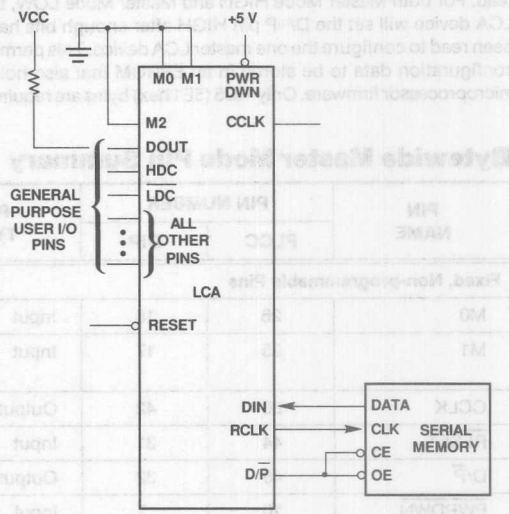


Figure 5a. Serial Mode

PIN NAME	PIN NUMBER		PIN TYPE	VALUE DURING CONFIGURATION	DESCRIPTION
	PLCC	DIP			
Fixed, Non-programmable Pins					
M0	26	18	Input	LOW	Mode Select
M1	25	17	Input	LOW	Mode Select
CCLK	60	42	Output	<Clock>	Configuration Clock
$\overline{\text{RESET}}$	44	31	Input	HIGH	Master Reset
D/$\overline{\text{P}}$	45	32	Output	LOW	Done/Program
PWRDWN	10	7	Input	HIGH	Power-down
User-Programmable Pins					
M2	27	19	Input	LOW	Mode Select
DIN	58	40	Input	<Data>	Configuration Data In
RCLK	57	39	Output	<Strobed>	SDPROM Clock
HDC	28	20	Output	HIGH	Constant "1" Level
$\overline{\text{LDC}}$	30	21	Output	LOW	Constant "0" Level

Figure 5b. Master Serial Configuration Mode Pin Summary

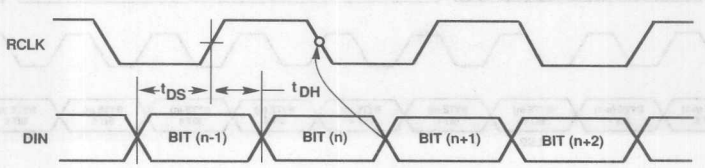


Figure 5c. Master Serial Mode Configuration Timing

Configuring Multiple LCAs

Recognizing that multiple LCAs could be used in a system, the designers added a feature which makes configuration easier by allowing several LCAs to be connected together in a daisy chain. In the Master mode, the first LCA reads data from the EPROM in parallel until it has received all of its configuration data. At this point its D/~P pin would normally be pulled passive HIGH but it is wire-ORed to the remaining LCA devices configured in the slave mode and is held LOW. The first LCA, although configured will not start to function, but continue to pass configuration data to the next LCA device. The data out line (DOUT) drives the DIN line of the first slave, and the CCLK input clocks the configuration data through until the next LCA is configured. The wire-OR action holds the D/~P input LOW preventing the configured LCAs from entering a functional mode. The sequence continues until the last LCA in the slave chain is configured and allows the D/~P line to go HIGH. All the configured devices in the chain start to function simultaneously because the composite wired-OR function goes HIGH on all LCAs simultaneously.

Due to the current sink capability of the D/~P input being able to handle only one pull-up load satisfactorily, only one of the LCA devices in the daisy chain should be configured with 'pull ups'. The first device in the chain can be configured in any of the four modes. All other devices in the chain would use the slave mode.

The daisy chain configuration mode is the easiest and most pin efficient, but the total configuration time increases linearly for each LCA added to the chain. It is also possible to configure LCAs in parallel in the slave or peripheral modes as shown in Figure 7. The total configuration time is then reduced to the configuration time of a single device.

Connecting parallel LCA's in the peripheral mode also allows the use of DMA transfer techniques to decrease configuration time.

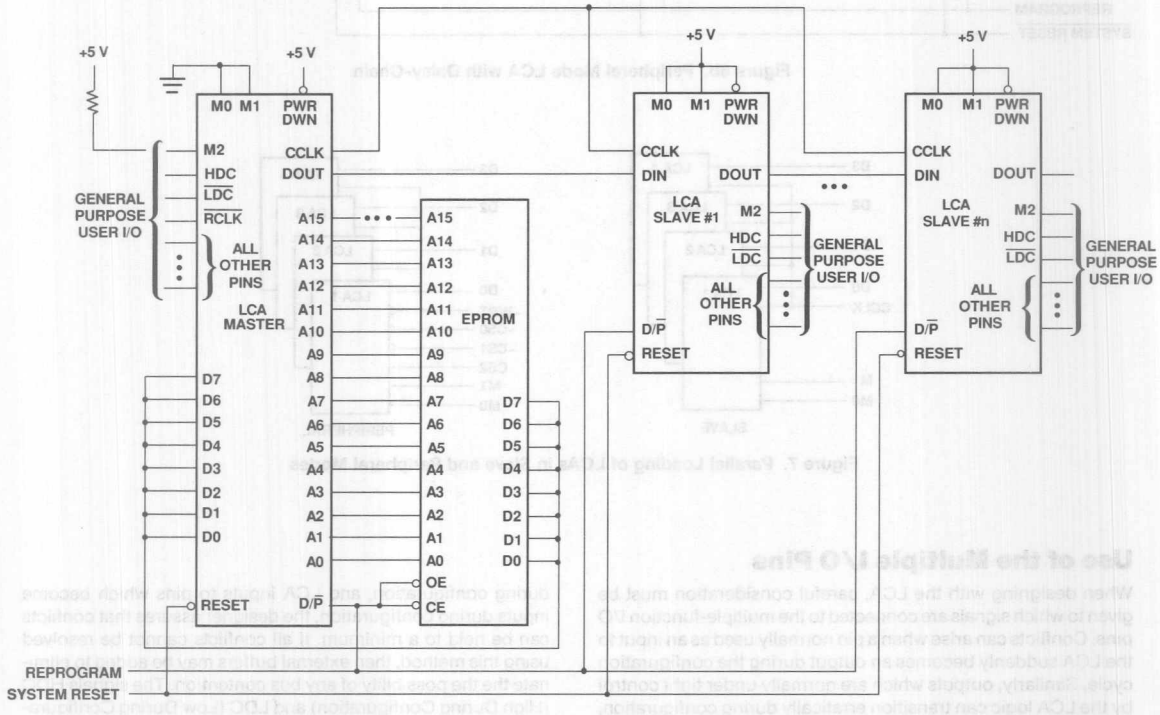


Figure 6a. Master Mode LCA with Daisy-Chain

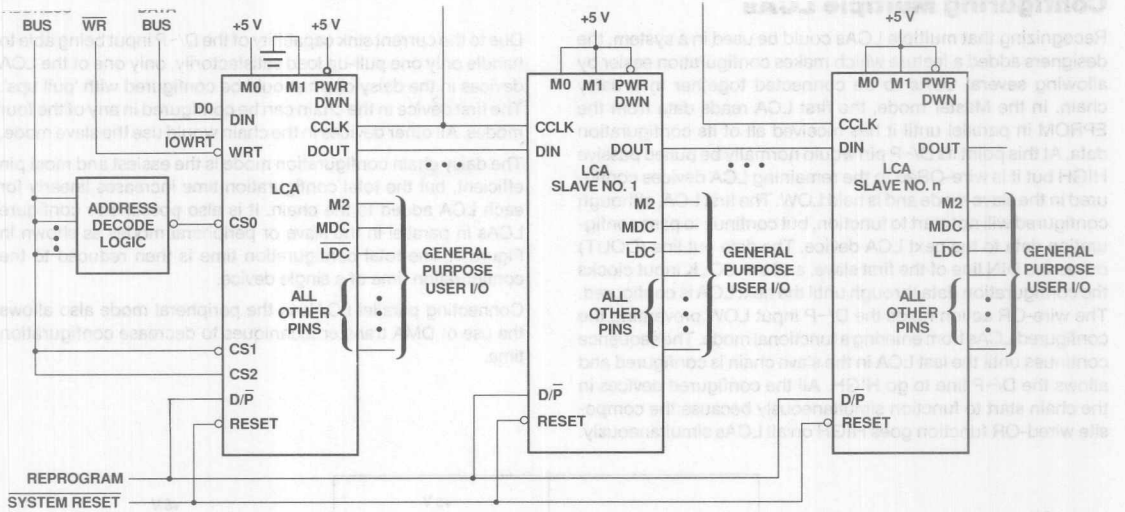


Figure 6b. Peripheral Mode LCA with Daisy-Chain

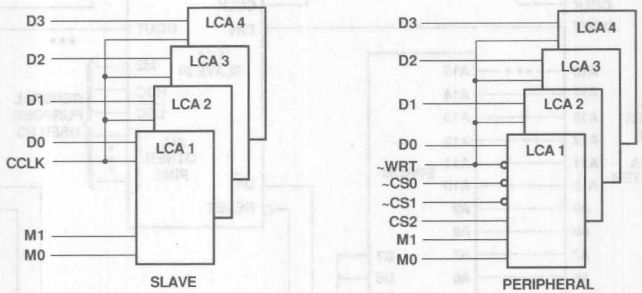


Figure 7. Parallel Loading of LCAs in Slave and Peripheral Modes

Use of the Multiple I/O Pins

When designing with the LCA, careful consideration must be given to which signals are connected to the multiple-function I/O pins. Conflicts can arise when a pin normally used as an input to the LCA suddenly becomes an output during the configuration cycle. Similarly, outputs which are normally under tight control by the LCA logic can transition erratically during configuration, causing adverse effects to the circuitry connected to them.

By simply assigning LCA outputs to pins which become outputs during configuration, and LCA inputs to pins which become inputs during configuration, the designer assures that conflicts can be held to a minimum. If all conflicts cannot be resolved using this method, then external buffers may be added to eliminate the the possibility of any bus contention. The signals HDC (High During Configuration) and LDC (Low During Configuration) can be used to enable or disable the buffers at the appropriate times. See Figure 8.

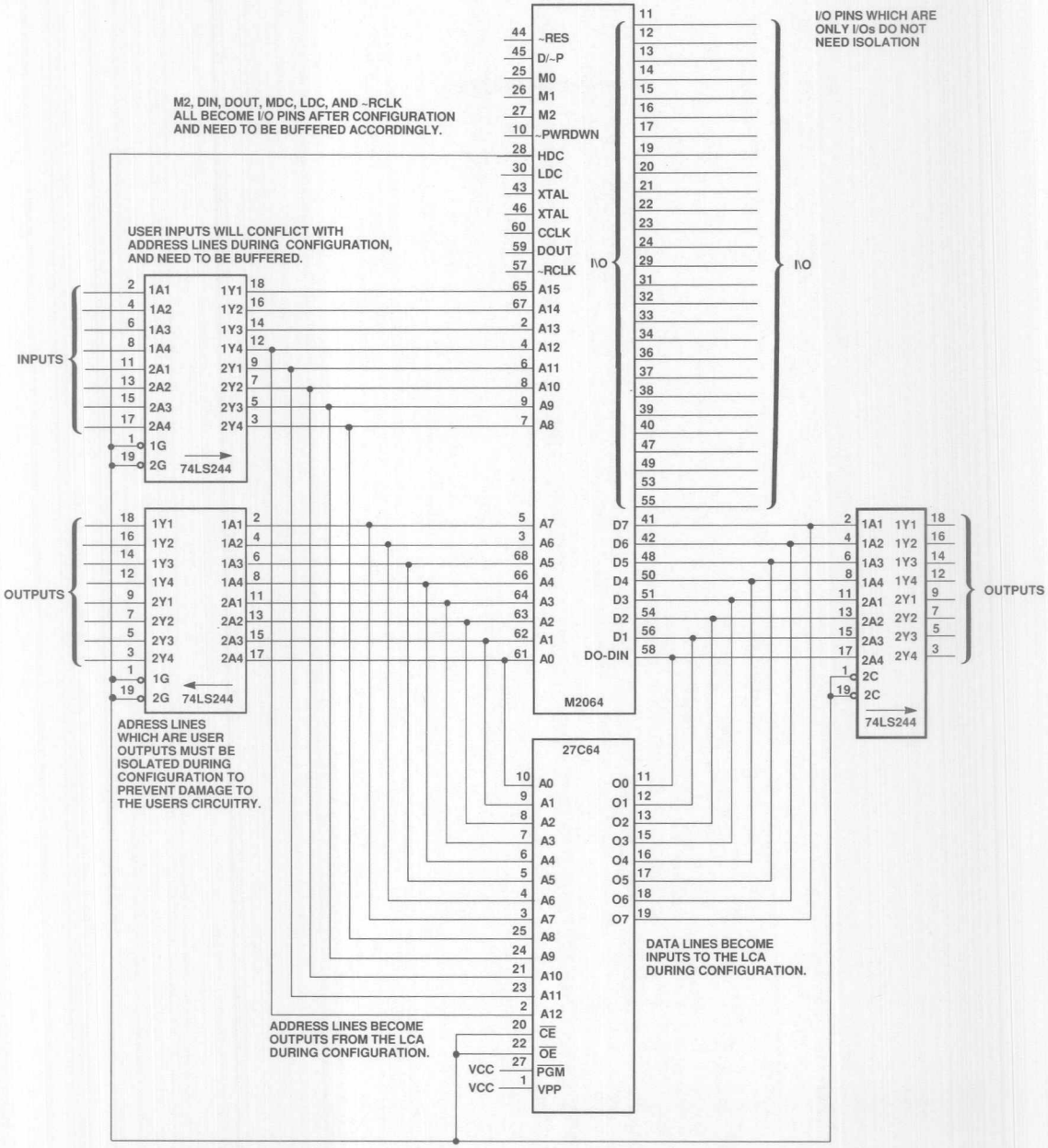
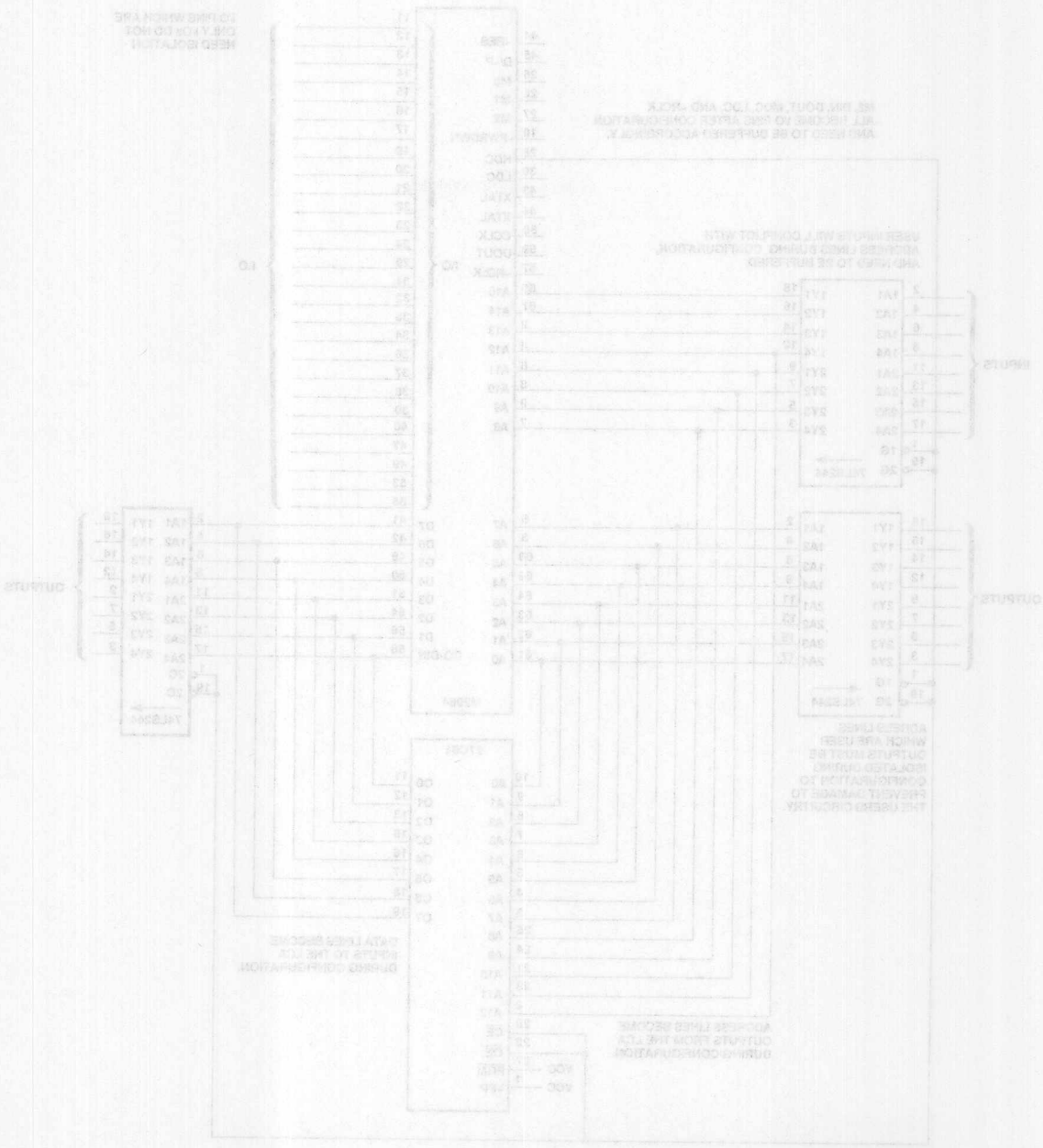


Figure 8. Isolation of I/O Pins During Master Mode Configuration



M2018 Provides Decoding for Six-Digit, Seven-Segment Liquid Crystal Display

Chris Jay

Abstract

The Logic Cell™ Array (LCA) from Monolithic Memories is a high density programmable device capable of supporting true VLSI logic functions. Its unique properties can be viewed as providing the benefits of Programmable Logic Devices (PLDs) while approaching the high functional density of a gate array. Combining these features makes the LCA device a product capable of bridging the gap between PLDs and Gate Arrays.

The LCA structure differs from the conventional concept of the PLD because it is based on a CMOS RAM cell architecture. Its internal logic circuits, input/output (I/O) resources plus the interconnect are all programmable. In comparison to the traditional fuse array of a PLD which is linked to a fixed logic structure. The efficiency of this logic structure reduces as functional density increases.

The low-power CMOS RAM locations of the LCA device must be initialized and configured immediately after power has been applied to the circuit. Once configured, functionality is maintained by the continued application of power to the device. To ease configuration, the device can reside alongside a low-cost EPROM which holds the configuration data. The LCA device can automatically load itself and be reprogrammed any number of times. This feature of reconfigurability can be used for system development, modifying existing designs, or supporting multiple system choice, with very little actual hardware modification. In many instances this can be achieved dynamically or "on the fly."

The following applications note describes the design of a six-digit, seven-segment decoder/driver for Liquid Crystal Displays (LCDs). Design methodology is outlined and the use of the XACT™ software is discussed in the role of a CAD design tool for the LCA device. The final designs are available to readers of this applications note on request.

2175 Mission College Blvd. Santa Clara, CA 95054-1592 Tel: (408) 970-9700 TWX: 910-338-2376
TWX: 910-338-2374
Monolithic Memories MMI

M2018 Provides Decoding for Six-Digit, Seven-Segment Liquid Crystal Display

Chris Jay

Introduction

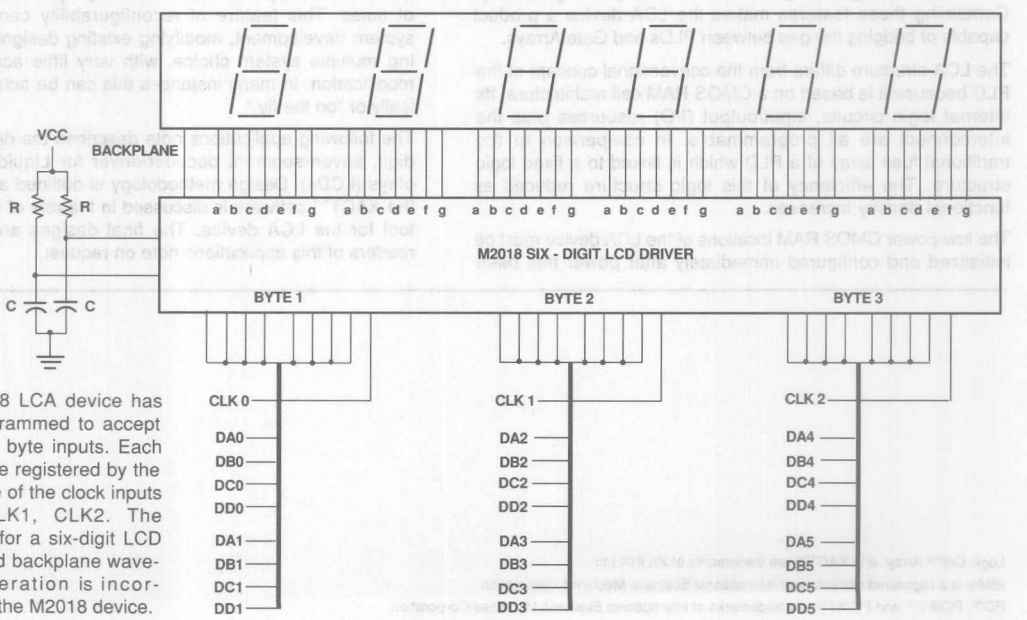
There are two types of low-voltage, seven-segment displays available as indicator panels of multimeters, frequency counters, tachometers and other digital instruments: the light emitting diode displays (LEDs), and Liquid Crystal Displays (LCDs). The advantages of the LCD are mainly associated with very low power consumption and ease of readability. Instruments and portable equipment can benefit from low-power CMOS technology and use these displays. The problem with LED displays is that a much higher operating current is required to illuminate the segments, virtually precluding their application in portable battery-operated equipment. Also, LED display outputs tend to "wash out" in sunlight, so they are very unsuitable in bright daylight, or where the ambient presence of light is high. In the display instrumentation of an automobile, for example, an LED display panel might be "washed out" by high levels of reflected sunlight. LCD displays are preferable because the visual quality is good even in brightly lit environments.

Despite the good visual quality and low-power consumption of the LCD display, an LED type can easily be driven from a multiplexed bus output. The output pin requirement of a seven-segment LED driver can be reduced considerably by a multiplexed output arrangement. In a six-digit display, each anode (or cathode) drive is selected in synchronism with its multiplexed seven-segment output. A total of thirteen pins are required, seven pins to drive the segments and six pins to control the anode (or cathode) of each display. If each digit is multiplexed at high speed with a proportionally larger pulse current per display, then the readability of the output is unaffected. With an LCD display, multiplexing is virtually impossible for two reasons. First, the LCD display has a slow response time and can not be multiplexed to give a good visual output. Second, the display backplane has to be pulsed. Any beating of a multiplexing frequency with the backplane bias frequency can occur, resulting in an unreadable output.

LCD displays have drawbacks because any driver circuit will need one pin per segment, plus a backplane driver pin for the whole display unit. Seven segments for each digit, in a six-digit display requires forty-two outputs pins, plus the one backplane pin. The input pin requirement is twenty-four, four pins for each of the six digits. The total pin count is sixty-seven. Additional input circuitry to support register enables and oscillator inputs would use up even more I/O resources and pins.

Block Function of the LCD Decoder/Driver

Figure 1 shows a schematic block diagram of the system as designed into the M2018 Logic Cell Array. Additional pins to synchronize the clocking of data into the internal data registers take three more pins. Since a backplane oscillator is required, two pins are configured to a resistor/capacitor (RC)



The M2018 LCA device has been programmed to accept three data byte inputs. Each byte can be registered by the rising edge of the clock inputs CLK0, CLK1, CLK2. The decoding for a six-digit LCD display and backplane waveform generation is incorporated in the M2018 device.

Figure 1. Liquid Crystal Display Decoder Driver

network. The RC combination generates a square wave which is used as a backplane bias to the display. The pin count is seventy-two.

Binary data applied to byte 1 is registered after the rising edge of CLK0. The registered data is decoded to illuminate the relevant segments forming the hexadecimal display. Decoded data is then routed to the two least significant digits of the six-digit LCD. Byte data applied to inputs 2 and 3 are loaded by a clock rising edge, CLK1 and CLK2, respectively. A backplane oscillator output drives the LCD's backplane with a low-frequency square wave, which is determined by the RC time constants of a resistor/capactitor network. To display a segment the decoder circuitry will drive the selected segment with a square wave that is 180 degrees phase-shifted from the backplane reference. If a segment is driven from an in-phase signal, then it is blanked.

The advantage of the M2018 Logic Cell Array in this application is that it is a CMOS device, and consumes low power at low operational frequencies. Its power consumption is dependant on the internal clocking frequency and also on the percent usage of on-chip logic. There are one hundred individually configured logic blocks in the M2018, and any logic blocks that remain unused will consume only static substrate leakage current. Since the M2018 contains seventy-four general-purpose user I/O blocks, there is ample programmable logic and I/O capability to perform decoding, registering and storage of data for six seven-segment digits.

LCA Configuration

Since the LCA device is based around a programmable array of volatile RAM cells, it must recieve configuration information at "power up." In this application it is programmed from an EPROM which holds the configuration data. Configuration takes place after power has been applied to the circuit, and it has been successfully reset. The LCA device enters a configuration mode, sequentially reads the data from the EPROM into its RAM cells, and becomes functionally operational in approximately twenty-five milliseconds. It disables the EPROM after configuration. The volatility of the M2018 is a distinct advantage in many applications. In this case, different display decode operations may be selected at "power up'. Higher order address lines may be "hardwired" to select different configuration patterns from separate pages in the EPROM. So a small amount of deferred design may be added to the system.

The M2018 has many advantages over custom VLSI circuits. Here, designers have control over how they wish to configure the circuit. Different display fonts can be programmed, for special characters. Most of the I/O pins can be reconfigured to ease printed circuit board layout.

Figure 2 shows the circuit connections of an 84-pin M2018 in a PLCC to an 8 Kbyte-wide EPROM. Approximately 2.5 Kbytes are required for configuration. Three configuration patterns may be cascaded into one 8 K x 8 EPROM, but in this application, two configuration patterns may be stored, one at base location of 0K and one at base location of 4 K. Address line A12 may be configured to a DIP switch or hardwire link, to select one of two configuration patterns. Two designs were developed, one for decoding binary data to a hexadecimal display and one for binary coded decimal. Both configuration patterns could be programmed into the EPROM and selected at a later time.

The configuration mode chosen for the LCA device was the Master Low mode. Data is sequentially read from the EPROM during the configuration cycle. After power has been applied and the RESET input has been deasserted, the M2018 will output a series of incrementing addresses to the memory starting at the initial address of 0000. The DONE/~PROG (D/~P), which is an output during configuration, is driven active LOW. It is tied to the CS and OE inputs of the EPROM. When configuration is finished, D/~P is pulled passive HIGH by an internally-configured "pull-up" resistor in the LCA device . The EPROM is then deselected. The normal data flow into the device is via dedicated input lines D0-D7 during configuration, and afterwards these become general-purpose I/O lines. The total time for configuration can vary between 17 to 34 ms.

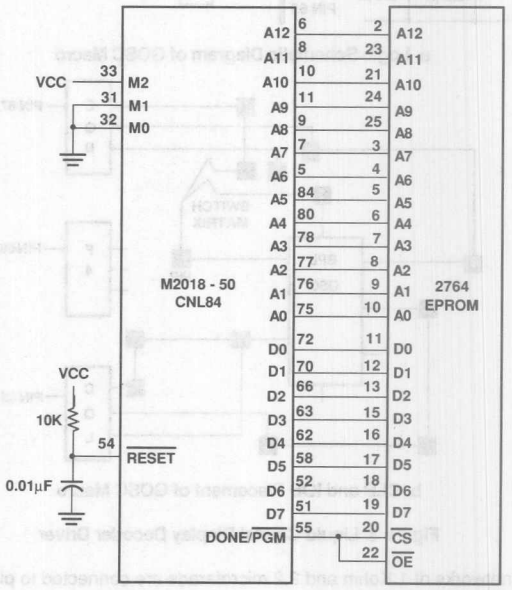


Figure 2. Liquid Crystal Display Decoder Driver

Design of the Backplane Oscillator

To establish a backplane bias frequency, an R/C network is used in conjunction with a logic block that is designed to function as a relaxation oscillator. The Configurable Logic Block (CLB) is connected to two Input Output Blocks (IOBs) which in turn are connected to two external R/C networks, R1,C1 and R2 and C2 as shown in Figure 3a. The calculation of the R/C values to create a low-frequency backplane oscillation of even mark space ratio is given below. For an even mark/space ratio R = R1 = R2 and C = C1 = C2. The time constants for period t are given by the following formula:

T1 = 0.35(C X R X 2) ;for TTL voltage ;thresholds.

T2 = 0.75(C X R X 2) ;for CMOS voltage ;thresholds.

The general expression for the calculation becomes:

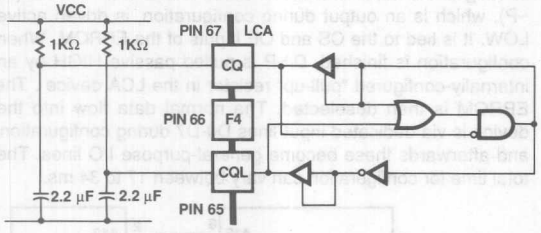
T = N X [(R1 X C1) + (R2 X C2)].....(1)

5

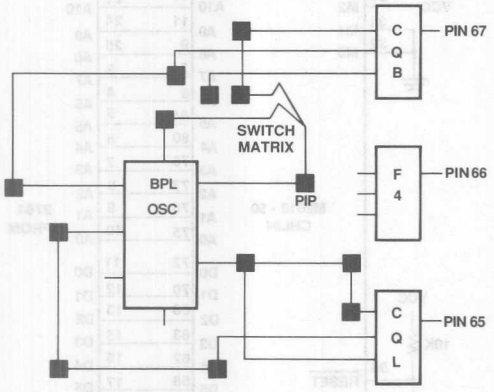
Where N is the TTL, or CMOS multiplying factor of 0.35 for TTL or 0.75 for CMOS. The LCA device can be programmed to be compatible with either technology.

The values of 1 Kohm and 2.2 microfarad gave a backplane oscillation frequency of about 80 Hz.

Figure 3a shows the schematic arrangement of the backplane oscillator as it would be configured in the LCA device. Two R/C



a. Logic Schematic Diagram of GOSC Macro



b. CLB and IOB Placement of GOSC Macro

Figure 3. Liquid Crystal Display Decoder Driver

networks of 1 Kohm and 2.2 microfarads are connected to pins 67 and 65 respectively. The placement of the blocks is shown in Figure 3b as it would be seen on the color monitor of a PC when the XACT Design Editor is used. Pins dedicated to configuration functions must not be loaded with components such as capacitors because they could prevent the LCA device from reading valid data. Pin 66, which is associated with data line D2 was therefore assigned to drive one of the LCD segments.

XACT Editor and Macro Support for Quick Design Entry

The logic design is accomplished by using the XACT Design Editor running in an IBM® PC/AT™ or PC/XT™. This support software allows users to edit their designs via a mouse interface and a keyboard input. The CLB and IOB elements may be configured and connected to form complete logic subsystems through the programmable interconnect. The XACT Editor allows the user to view a section of the LCA device on the display unit of the PC, for CLB and IOB placement, and distribution of interconnect. Supported by the zoom-in and zoom-out facility, and a "world view" feature (this allows the user to see the entire LCA device layout) enables a designer to "hook" subsystems of the design together. Menus are displayed at the head of the video display, and a designer may move the cursor with a mouse to select options. These include BLOCK commands enabling CLB/IOB editing; CONFIG to enable configuration of blocks; and NET and PIN to assign pins to nets, so creating the overall interconnection. This could be analagous to interconnections on a printed-circuit card, making the electrical connection between pins of 74LSXX or other types of logic devices.

The XACT editor is equipped with a support MACRO library. This library allows widely used functions to be invoked without redesign. The relaxation oscillator, used for the backplane generator, is supplied as an existing design configuration in the MACRO. The oscillator MACRO may be called by downloading the GOSC MACRO file.

When using the XACT editor the application of already existing MACRO support files can dramatically reduce the overall design cycle time. The six-digit decoder display driver is basically repetition of the same six logic subsystems, one for each digit as shown in Figure 4. The CUTMACRO feature is useful as a support facility in avoiding repetitive design operations on individual IOBs or CLBs. Once a CLB has been designed, it can be stored as a MACRO, and given a user-defined name. The designer may call that MACRO wherever the IOB or CLB needs configuration and placement. The CUTMACRO feature also supports multiple CLBs, IOBs and interconnects of subsystems. A decode section identical to the one shown in Figure 4 can be stored as a MACRO and given a user-defined name. The design for a decoder section needs only to be performed once. The remaining five sections may be called and placed using the user-defined MACRO support facility.

Widely Used XACT Functions

As described,the XACT software is menu-driven and operations supported by the design editor are listed at the head of the screen with a schematic representation of a portion of the device shown below. A mouse interface permits the designer to scan a cursor over the schematic, or choose a functional operation by "clicking" on the selected option. Following are the available options from left to right:

NET PIN BLK CONFIG SCREEN MISC PROFILE

When selected, a heading will list its menu options. The BLK command gives nine options. The most commonly used options are EDITBLK and ENDBLK, which allow the designer to select a specific logic block and enter a logic description into it. The latter command will terminate the editing process and return the designer to a world view of the LCA schematic. Details on editing CLBs are described in more detail in the CLB Configuration Section. Other commonly used commands in the BLK menu include COPYBLK, to copy a repetitive logic design into other unconfigured blocks, DELBLK to erase unwanted designs from the network; CLRBLK, to clear the contents of a block but leave its input and output pins attached to the nets; MOVBLK, move a configured block with its associated nets to another block location; and NAMEBLK, to permit the assignment of user-defined names. Each time a function is invoked, the XACT editor prompts for a response at the bottom of the screen. For example, EDITBLK would invoke a "SELECT

BLOCK" invitation. The user then "clicks on" the block he/she wishes to configure.

The CONFIG command has six subcommands which relate to the editing of a CLB. The BASE command directs which logic arrangement is to be selected: one Boolean function of four-input variables, or two functions of three-input variables, or multiplexed input selecting either one choice of two three-input variables. EQUATE is commonly used, and allows a designer to generate a Boolean equation from keyboard input and configure it to an ouput. Other choices include ORDER, to establish blank truth tables for Karnaugh map entry; CLEAR, to delete unwanted functions; and CDATA (see data), for a text description of the block configuration.

The creation of configured blocks must have the support of interconnections. The NET and PIN commands are invoked to establish the connection of CLB outputs to CLB or IOB inputs. The NET menu may be invoked to create a net onto which input pins and one output pin is listed. The ADDNET command allows a designer to add a net to the netlist, and ADDPIN allows the user to add pins to a selected net. The NAMENET subcommand in the NET menu allows the default value of netx (where x is an XACT default-assigned integer) to be overruled by a user-assigned name. This is similar to the NAMEBLK command for BLK functions. Nets can be routed, unrouted, deleted and merged by the ROUTE, UNROUTE, DELNET and JOINNET commands, respectively. Manual editing of the net is invoked by the EDITNET command. A net is established and manually interconnected to Programmable Interconnect Points (PIPs) to create a circuit. The other NET options include FLAGNET, which assigns critical or non-critical status to a net; and HIGHLIGHT/UNHIGHLIGHT commands which "bright up" or "clear" net distribution on the "world view."

The PIN commands are: ADDPIN for adding pins to nets, and CLEARPIN for removing them. The SWAPPIN allows pins to be switched without switching functionality, where SWAPSIG exchanges the logic functionality inside the CLB without switching the pins. MOVEPIN will move a pin from one location to another, and ROUTEPIN will establish an interconnection on a net.

These and other menu functions are described in greater detail in Monolithic Memories' XACT Development System manual.

Design Methodology

Figure 4 shows a block schematic of one seven-segment decoder driver. Seven logic blocks decode the four data inputs DA, DB, DC, and DD, where DA is the least significant data input. Each block has the capability of storing the data in a register inside the logic block. The backplane oscillator drives the LCD backplane for all segments. The individual segments are driven from an exclusive-'OR" (XOR) gate, which is programmed to invert the backplane waveform for an illuminated segment, and not invert for a blanked segment. The segment decoder circuits, SA0, SB0 to SG0, provide a logic LOW output for an active segment drive. Therefore, the backplane input to the XOR gate is inverted in a second logic block and provides the correct phase control for the segment driver. The configuration was repeated six times in the LCA device to support a total of six digits.

Figure 5 shows the truth table for decoding four binary inputs into a hexadecimal segment drive. DA is the least significant binary input and DD is the most significant. The hexadecimal weighting of the binary input is also shown in the table. To illuminate a zero digit, all the segments are driven active except for segment "g," for digit "one"; segments "b" and "c" are active, and for digit "two" segments "a," "b," "d," "e" and "g" are driven. The complete decode arrangement is shown at the head of Figure 6.

The Karnaugh maps, derived from the truth table shown in Figure 5 are given in Figure 6, and provide the decode equations for all the segments. Logic "one" entries represent illuminated segments and logic zeros represent extinguished segments. A greater reduction efficiency of minterm entries was achieved using reduction techniques applied to map locations containing logic zeros. This required an inverted input to the exclusive OR (XOR), polarity control gate, as shown in Figure 4. The logic equations derived from the

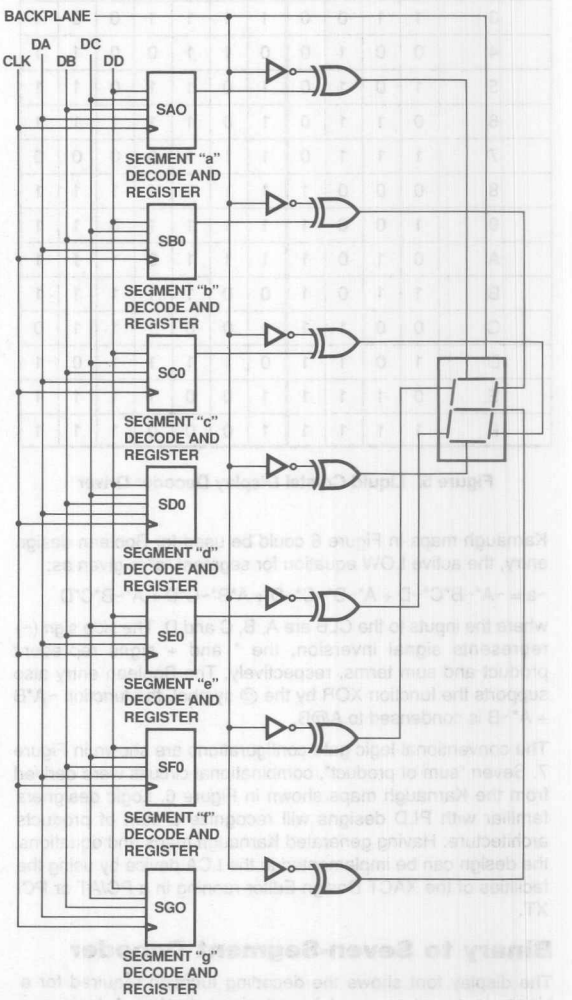
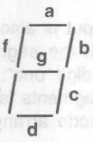


Figure 4. Liquid Crystal Display Decoder Driver

5



BINARY ENCODED INPUT					SEGMENT LABEL						
	LSB			MSB							
HEX.	DA	DB	DC	DD	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	1	0	0	0	0	1	1	0	0	0	0
2	0	1	0	0	1	1	0	1	1	0	1
3	1	1	0	0	1	1	1	1	0	0	1
4	0	0	1	0	0	1	1	0	0	1	1
5	1	0	1	0	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	1	1	1	0	1	1	1	0	0	0	0
8	0	0	0	1	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
A	0	1	0	1	1	1	1	0	1	1	1
B	1	1	0	1	0	0	1	1	1	1	1
C	0	0	1	1	1	0	0	1	1	1	0
D	1	0	1	1	0	1	1	1	1	0	1
E	0	1	1	1	1	0	0	1	1	1	1
F	1	1	1	1	1	0	0	0	1	1	1

Figure 5. Liquid Crystal Display Decoder Driver

Karnaugh maps in Figure 6 could be used for Boolean design entry, the active LOW equation for segment "a" is given as:

~a = ~A*~B*C*~D + A*~B*~C*~D + A*B*~C*D + A*~B*C*D

where the inputs to the CLB are A, B, C and D. The tilde sign (~) represents signal inversion, the * and + signs represent product and sum terms, respectively. The Boolean entry also supports the function XOR by the @ symbol, the function ~A*B + A*~B is condensed to A@B.

The conventional logic gate configurations are shown in Figure 7. Seven "sum of product", combinational circuits were derived from the Karnaugh maps shown in Figure 6. Logic designers familiar with PLD designs will recognize a sum of products architecture. Having generated Karnaugh maps and equations, the design can be implemented in the LCA device by using the facilities of the XACT Design Editor running in a PC/AT or PC/XT.

Binary to Seven-Segment Decoder

The display font shows the decoding function required for a binary to seven-segment hexadecimal display. A logic one represents an illuminated segment, from the table shown in Figure 5.

LCD Seven-Segment Display Driver in LCA

D,C \ B,A	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	C	D	F	E
10	8	9	B	A

For binary input ABCD the hexadecimal value of that binary weighting is entered in the correct location of the Karnaugh map.

D,C \ B,A	00	01	11	10
00	1	0	1	1
01	0	1	1	1
11	1	0	1	1
10	1	1	0	1

~a = ~A*~B*C*~D + A*~B*~C*~D + A*B*~C*D + A*~B*C*D

(a)

D,C \ B,A	00	01	11	10
00	1	1	1	1
01	1	0	1	0
11	0	1	0	0
10	1	1	0	1

~b = ~A*B*C + A*B*D + ~A*C*D + A*~B*C*~D

(b)

D,C \ B,A	00	01	11	10
00	1	1	1	0
01	1	1	1	1
11	0	1	0	0
10	1	1	1	1

~c = ~A*C*D + B*C*D + ~A*B*~C*~D

(c)

D,C \ B,A	00	01	11	10
00	1	0	1	1
01	0	1	0	1
11	1	1	0	1
10	1	1	1	0

~d = A*B*C + ~A*B*~C*D + ~A*~B*C*~D + A*~B*~C*~D

(d)

D,C \ B,A	00	01	11	10
00	1	0	0	1
01	0	0	0	1
11	1	1	1	1
10	1	0	1	1

~e = A*~B*~C + A*~D + ~B*C*~D

(e)

D,C \ B,A	00	01	11	10
00	1	0	0	0
01	1	1	0	1
11	1	0	1	1
10	1	1	1	1

~f = B*~C*~D + A*~C*~D + A*B*~D + A*~B*C*D

(f)

D,C \ B,A	00	01	11	10
00	0	0	1	1
01	1	1	0	1
11	0	1	1	1
10	1	0	1	1

~g = ~B*~C*~D + A*B*C*~D + ~A*~B*C*D

(g)

Figure 6. Liquid Crystal Display Decoder Driver

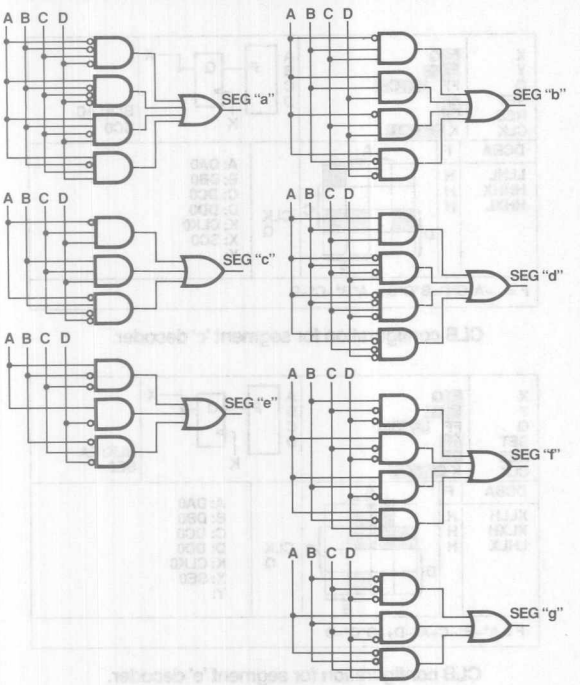


Figure 7. Liquid Crystal Display Decoder Driver

The decode logic for each segment is shown in the conventional sum of products form and was derived from the Karnaugh Maps. It is possible to encode these logic configurations in to one "CLB" for each segment. The "CLB" is capable of producing one output that is a Boolean function of up to a maximum of four inputs.

Editing the Design

The XACT Design Editor was used to create the design, and generate the configuration data, suitable for programming into an EPROM. Three programmable elements have to be considered; the IOBs, CLBs and the interconnection of IOB and CLB blocks.

CLB Configuration

Figure 8 shows the exploded view of a CLB that has been configured as the segment "a" decoder. The XACT Design Editor provides a mix of text and diagram editing supported by a keyboard and mouse interface. The logic equation is shown as text entry at the bottom of the diagram. The combinational block, AA (user defined as SA0 by invoking the NAMEBLOCK feature in XACT), has been configured as the "a" segment decoder. Each CLB can be configured as a single output function of four Boolean input variables, or two output functions of three-input variables each. Both configurations were used in this design example.

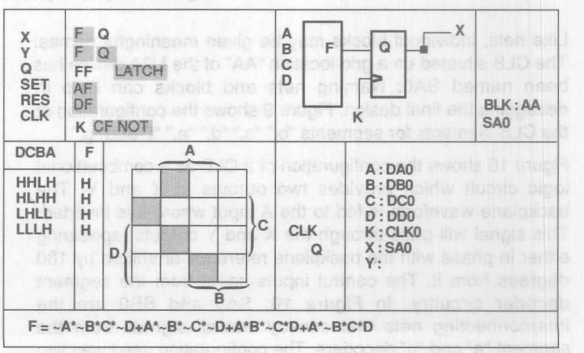
The Karnaugh map reflects the equation entered from the keyboard. An alternate way of entering configuration information may be achieved by using the mouse to select individual locations in the Karnaugh map. These locations may be "clicked" on or off, to select or deselect minterm entries. The equation relevant to the Karnaugh map entry is updated after modification. If designers have generated Karnaugh maps during the design cycle of the project, they could use this visual aid to enter and check the logic integrity of the design. Also, a truth table is available for verification purposes.

An unconfigured CLB will not display a Karnaugh map so to generate a four-input map, the instruction:

Config(Order(F(A(B(C(D)))))

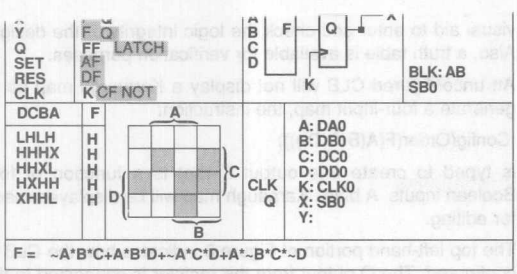
is typed to create one output F that is a function of four Boolean inputs. A blank Karnaugh map will be displayed ready for editing.

The top left-hand portion of Figure 8 indicates how the CLB is configured. The Q output from the register is connected to the X output of the CLB. To activate this path, the mouse is used to select the Q option. The register could be bypassed if the F output was selected. Either register or combinational configurations may be realized. The path from the register output to the CLB X output is established by activating the PIP represented by the block tying Q to X shown in Figure 8. The alternate output, Y has the same assignment options of registered or combinational outputs. There are two configurations available to the storage element; either register or transparent latch. The register's output will change as a result of a transition being applied to its clock input. For a latch, the control input is level-sensitive requiring a logic HIGH or LOW level to distinguish between storage or a transparent mode of operation. In the CLB the clock input polarity can be selected by the NOT function in the CLK submenu. Two features that were not used in this design are the asynchronous SET and RESET functions. If required, the SET option can be invoked by selecting either the A input, or choosing a combinational output from F, which can also perform RESET. A single input to RESET the register or latch is the D input. To the bottom right of the CLB configuration diagram in Figure 8, all the inputs to, and outputs from, the CLB are listed. The net to which each CLB input or output is associated is shown after the colon. The A input is tied to the net DA0; B, C and D inputs are connected to nets DB0, DC0, and DD0, respectively. The user can assign meaningful names to nets in the design process. In this example, net SA0, is segment "a" of the least significant digit and is driven from the X output of the CLB. DA0-DD0 are the four binary inputs for the least significant digit. The clock input is assigned to the K terminal of the CLB, thus, the decoded output may be registered after a rising edge of the clock has been applied to the net CLK0.

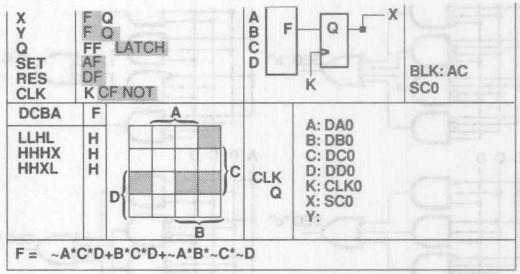


CLB configuration for segment 'a' decoder.

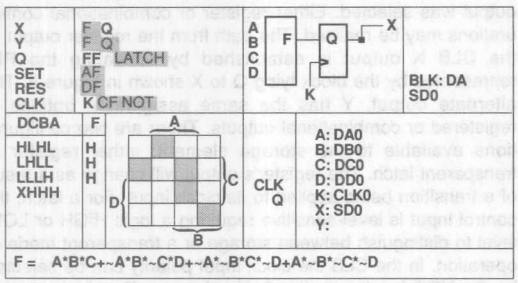
Figure 8. Liquid Crystal Display Decoder Driver



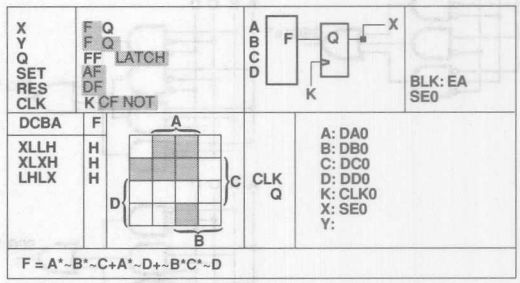
CLB configuration for segment 'b' decoder.



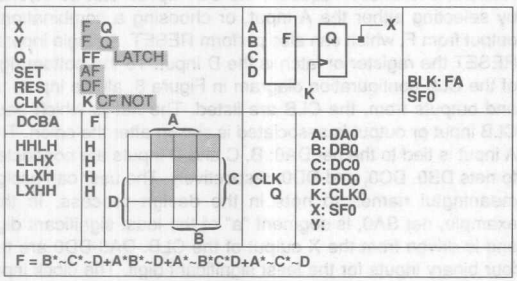
CLB configuration for segment 'c' decoder.



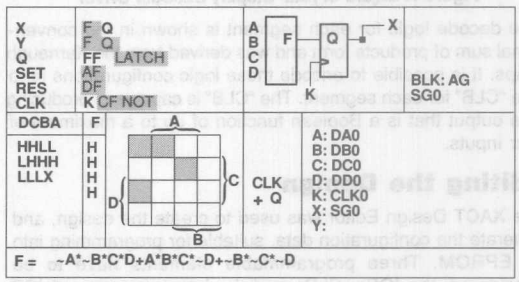
CLB configuration for segment 'd' decoder.



CLB configuration for segment 'e' decoder.



CLB configuration for segment 'f' decoder.



CLB configuration for segment 'g' decoder.

Figure 9. Liquid Crystal Display Decoder Driver

Like nets, individual blocks may be given meaningful names. The CLB situated on a grid location "AA" of the LCA device has been named SA0. Naming nets and blocks can help in debugging the final design. Figure 9 shows the configuration of the CLB elements for segments "b," "c," "d," "e," "f" and "g."

Figure 10 shows the configuration of a CLB as a combinational logic circuit which provides two outputs at X and Y. The backplane waveform is fed to the A input where it is inverted. This signal will pass through the X and Y outputs, appearing either in phase with the backplane reference or shifted by 180 degrees from it. The control inputs come from the segment decoder circuitry. In Figure 10, SA0 and SB0 are the interconnecting nets that convey control signals from the segment "a" and "b" decoders. The configuration used was two Boolean outputs as functions of three-input variables, so outputs F and G were selected.

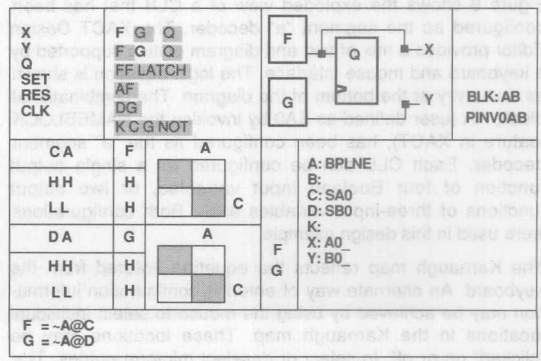
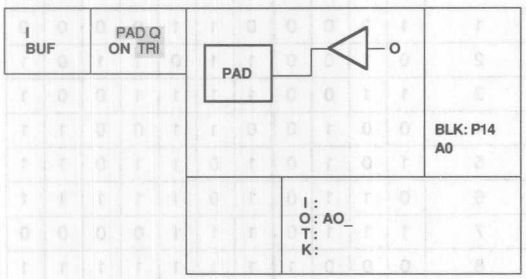


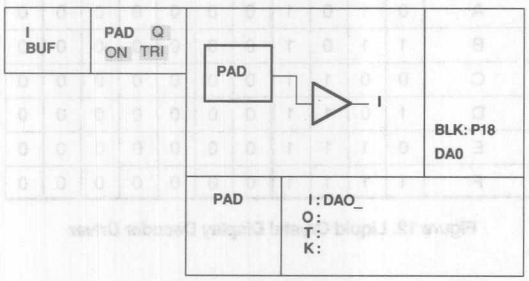
Figure 10. Liquid Crystal Display Decoder Driver

IOB Configuration

The Input/Output blocks must be configured to drive the segments, backplane, and to receive data for decoding. The segment and backplane IOBs are configured as outputs. Inputs are configured for data ports DA0-DD0, DA1-DD1, up to DA5-DD5. The IOBs are considerably less complex than the CLBs, having interface functionality rather than Boolean logic functionality. Each IOB is capable of being programmed as input, output, bidirectional, with HIGH-Z output, and registered input. Individual or composite I/O functions may be selected. Figure 11a shows an output buffer, the input of which is driven from net A0_. Net A0_ is driven from the X output of the CLB featured in Figure 10. The designer has some control over the placement of IOBs around the perimeter of the LCA device, and can use the BLKMOVE commands to optimize pad layout for the best printed circuit design. Figure 11b shows an IOB that has been configured to receive data. The buffer is an input driving net DA0_.



a. Liquid Crystal Display Decoder Driver



b. Liquid Crystal Display Decoder Driver

Figure 11. Input/Output Block Configuration

The IOB has been configured as an output buffer to drive the segment "a" of the least significant digit. The block has been assigned the name A0. the output buffer has been turned on and the input to that buffer has been configured to net AO_. All the segments and the backplane of the LCD have been assigned an output buffer.

The data path to the LCA device for binary data is via IOBs configured as input buffers. The binary inputs DA0, DB0, DC0, and DD0 for the least significant digit and the configured IOBs are assigned user-defined names DA0, DB0, DC0, and DD0. The input nets are DA0_, DB0_, DC0_, and DD0_ respectively. This is repeated for digits 1, 2, 3, 4, and 5.

XACT Supports Design Rule Checking

During the design of the LCD decoder driver circuit the Design Rule Checker (DRC), was invoked to verify that no fundamental design rules were being violated. Errors can easily occur, especially for the uninitiated user. Errors such as two CLB outputs driving one net are usually caught dynamically. Also, a net listing of inputs that are not driven by an output would constitute an error. A general DRC run will list all design violations, errors and warnings. Warnings such as an assigned CLB output that is not connected to a net. This information can be sent to a line printer and the resulting list of Errors and Warnings can be used for design correction. Invoking DRC will check blocks, interconnect and nets. Also, prior to using the MAKEBITS command (for the eventual MAKEPROM) software, the DRC is invoked to trap any possible design violations. Of course, the design rule checker does not screen the design for logic function integrity. An additional software package is available in P-SILOS™ as a logic, and timing simulator. Inputs may be activated by HIGH, LOW, HIGH-Z levels etc., while output logic levels may be listed during the simulation run.

Conclusion

The design was modified for a decimal decoder display driver. Figure 12 shows the modifications necessary to each segment decoder. The choice of a latched version of the display driver was developed as an alternative to the registered type. In each registered CLB, the option was changed for a latched option. By removing the backplane oscillator and making the net BPLNE an input which is driven HIGH or LOW, high efficiency LED displays of common anode or cathode may be driven. The designs developed are as follows:

XDES01.LCA REGISTERED HEX DECODER DRIVER.

XDES02.LCA REGISTERED DEC DECODER DRIVER.

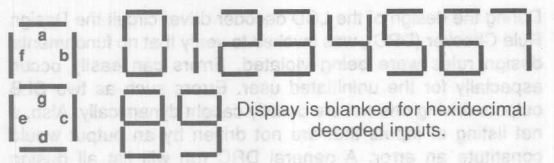
XDES03.LCA LATCHED HEX DECODER DRIVER.

XDES04.LCA LATCHED DEC DECODER DRIVER.

These designs are available as bit patterns for programming EPROMs on request.

The Print World of the design DES01.LCA shows the placement of CLBs, IOBs and the routing for the final design. This can be used as a reference for wiring the device into a circuit. Pin 1 is shown at the top center of the diagram and is the GND connection. The pins are numbered counter-clockwise from that reference pin. For example DB3 is connected to pin 2 and D2 connected to pin 3 etc.

LCD Seven-Segment Display Driver in LCA Binary Coded Decimal



Display is blanked for hexidecimal decoded inputs.

D,C \ B,A	00	01	11	10
00	1	0	1	1
01	0	1	1	1
11	0	0	0	0
10	1	1	0	0

~a = C*D + B*D + ~A*~B*C*~D + A*~B*~C*~D

(a)

D,C \ B,A	00	01	11	10
00	1	1	1	1
01	1	0	1	0
11	0	0	0	0
10	1	1	0	0

~b = C*D + B*D + ~A*B*C* + A*~B*C

(b)

D,C \ B,A	00	01	11	10
00	1	1	1	0
01	1	1	1	1
11	0	0	0	0
10	1	1	0	0

~c = C*D + B*D + ~A*B*~C

(c)

D,C \ B,A	00	01	11	10
00	1	0	1	1
01	0	1	0	1
11	0	0	0	0
10	1	1	0	0

~d = C*D + B*D + A*B*C + ~A*~B*C + A*~B*~C*~D

(d)

D,C \ B,A	00	01	11	10
00	1	0	0	1
01	0	0	0	1
11	0	0	0	0
10	1	0	0	0

~e = A + ~B*C + B*D

(e)

D,C \ B,A	00	01	11	10
00	1	0	0	0
01	1	1	0	1
11	0	0	0	0
10	1	1	0	0

~f = B*D + A*B + C*D + B*~C + A*~C*~D

(f)

D,C \ B,A	00	01	11	10
00	0	0	1	1
01	1	1	0	1
11	0	0	0	0
10	1	1	0	0

~g = C*D + B*D + A*B*C + ~B*~C*~D

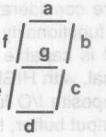
(g)

Karnaugh maps for binary coded decimal drivers. A logic zero represents an extinguished segment.

Figure 12. Liquid Crystal Display Decoder Driver—For Binary Coded Decimal Output

BCD to Seven-Segment Decoder

The table below shows the decoding function required for a BCD to seven-segment hexadecimal display. A logic one represents an illuminated segment.



BINARY ENCODED INPUT					SEGMENT LABEL						
	LSB			MSB							
HEX.	DA	DB	DC	DD	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	1	0	0	0	0	1	1	0	0	0	0
2	0	1	0	0	1	1	0	1	1	0	1
3	1	1	0	0	1	1	1	1	0	0	1
4	0	0	1	0	0	1	1	0	0	1	1
5	1	0	1	0	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	1	1	1	0	1	1	1	0	0	0	0
8	0	0	0	1	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
A	0	1	0	1	0	0	0	0	0	0	0
B	1	1	0	1	0	0	0	0	0	0	0
C	0	0	1	1	0	0	0	0	0	0	0
D	1	0	1	1	0	0	0	0	0	0	0
E	0	1	1	1	0	0	0	0	0	0	0
F	1	1	1	1	0	0	0	0	0	0	0

Figure 12. Liquid Crystal Display Decoder Driver

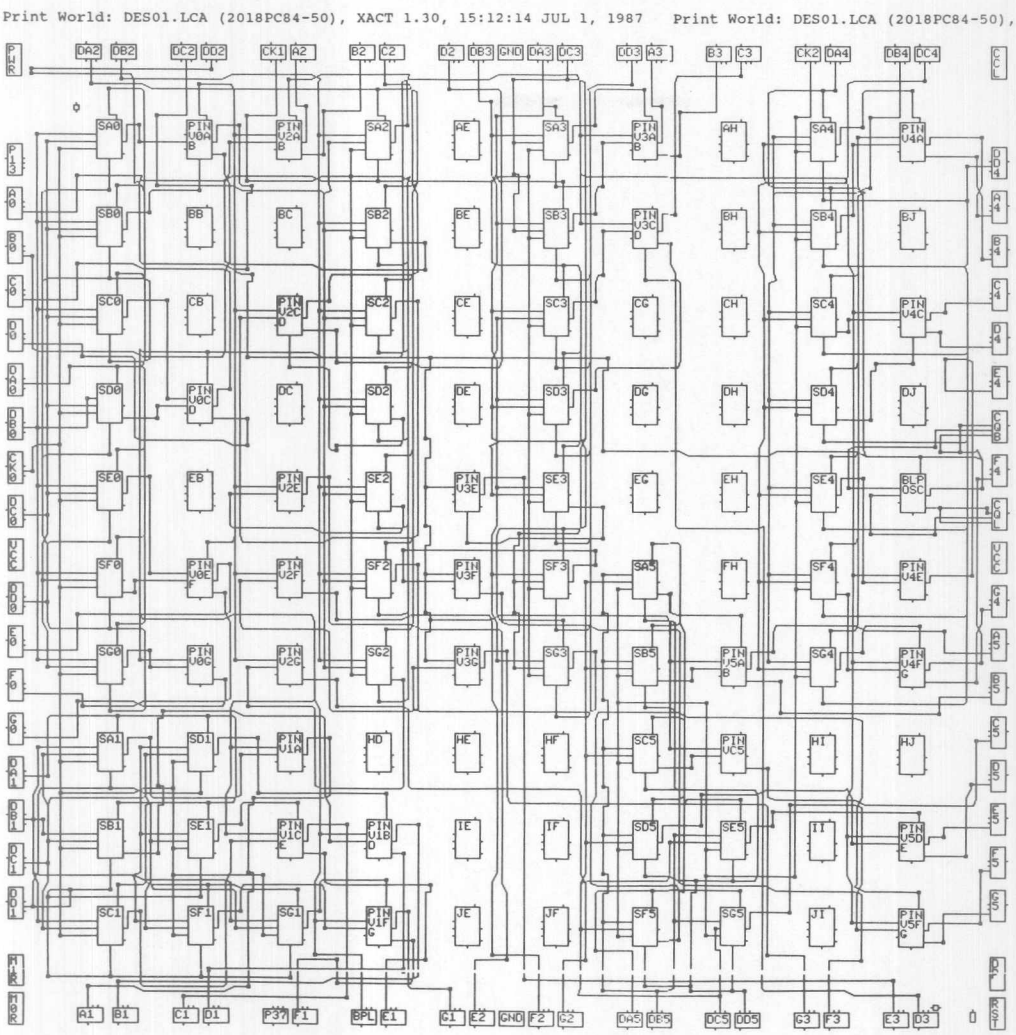


Figure 12. Liquid Crystal Display Decoder Driver

LCA Counter Applications

Chris Jay and Karen Spesard

Abstract

Counters are used in many logic systems to control and synchronize events. Essentially, counters have one thing in common, they are all state machines. State machines, unlike basic combinational functions, require registers with feedback. The design creates an output that is a function of the previous state of the registers in the machine, and in some cases a function of other combinational inputs as well. Listed below are some of the systems that would use various types of counters.

- Direct Memory Access (DMA) Controllers
- Video RAM Refresh
- Dynamic RAM Control Refresh Counters
- Event timing
- Sequence Controllers
- Disc Controllers
- Status Sequencers

The type of counter chosen will depend upon the application. The simplest type of counter would be a free-running type as used for refreshing DRAM or Video RAM. Counter types vary, depending on application from the simple free-running type to loadable binary up/down counters which would be used in applications such as DMA control.

To design efficient counters in the LCA device, designers must consider the desired performance while keeping in mind the available logic and interconnection resources.

Familiarization with the different counter types and their characteristics will enable designers to choose the best counter for a specific application.

5

LCA Counter Applications

Chris Jay and Karen Spesard

Introduction

The LCA device has an architecture that is very well suited to the development of some counters. Designers already familiar with state machine applications in Programmable Logic Devices (PLDs) will probably be familiar with how to design counters in Programmable Array Logic (PAL®) and/or Programmable Logic Arrays (PLA). The differences between the structure of the PLA or PAL device and the Monolithic Memories' Logic Cell™ Array (LCA) impose both restrictions and freedoms in certain types of state machine design.

Ample width of the PLD "AND" gate means that counter depth can be limited only by the number of registers in the device. This applies especially in the PAL24X family that is specifically designed for applications in high-performance binary counting. In the LCA device, the maximum number of Boolean inputs to any CLB is four. This limiting factor is compensated by the high number of CLBs present in the device architecture. The design philosophy used in an LCA device for developing state machines is different from that of a PLD, but not less effective. Parallel counter architecture, and lookahead-carry techniques can be employed to create machines capable of medium performance.

Counters Using Shift Registers

The type of state machine well suited to an LCA device is based on a shift register. These types may be subdivided into different categories. In each state machine, there are "n" general registers, and the number of states that can be reached vary with counter type. The number of useable states are listed below with the counter type:

1) Johnson counters, 2n, states.
2) Linear feedback shift registers, with $2^n - 1$ states.
3) Modified linear feedback shift register, with 2^n states.

These state machines are based on shift registers using the absolute minimum amount of feedback, and this is applied only to the least significant register in the chain. The CLBs may be located at close proximity in the LCA device and benefit from the smallest propagation delays provided by local interconnection. The disadvantage of the shift register structure is the inability to count through the conventional binary sequence. This might or might not be of any concern, depending on the system application.

The Johnson Counter

All of the states of a six-bit Johnson counter are listed in Table 1. The "D"-type register chain shown in Figure 1 illustrates the schematic diagram of the circuit. The output from Q0 feeds directly to D1, and Q1 to D2, and so on. If these registers are configured in adjacent CLBs, direct interconnect can be used to link each one. The output of the final register Q5 is inverted and fed back to the D0 input of the first register. If the entire structure is configured in a column or row in the LCA device, a long line should be used to convey the feedback signal back to the D0 input of the least significant register. Minimum propagation delay is achieved by using long line interconnect. The advantage gained with the small propagation delay of direct short interconnections between adjacent CLBs is not lost by the use of this long feedback path.

The counter design shown in Figure 1 uses six registers and the maximum number of states reached is 12, two times the number of registers. A binary counter with six registers could reach sixty-four individual states but would need combinational feedback to each register. Feedback propagation delay could degrade the potential clocking speed of the counter. The Johnson counter, with its decreased number of states, should be used whenever maximum operational performance is needed.

STATE	Q0	Q1	Q2	Q3	Q4	Q5	HEX
0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	1
2	1	1	0	0	0	0	3
3	1	1	1	0	0	0	7
4	1	1	1	1	0	0	F
5	1	1	1	1	1	0	1F
6	1	1	1	1	1	1	3F
7	0	1	1	1	1	1	3E
8	0	0	1	1	1	1	3C
9	0	0	0	1	1	1	38
10	0	0	0	0	1	1	30
11	0	0	0	0	0	1	20

Table 1. Truth Table of the Johnson Counter Counter Applications

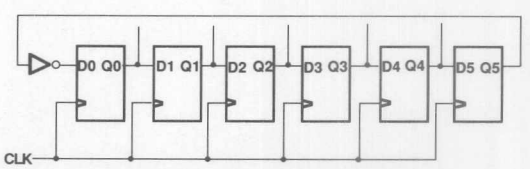


Figure 1. Johnson Counter

Linear Feedback Shift Register

Figure 2 shows a modification to the Johnson counter. In this three-bit counter, six states are reached. The eXclusive NOR feedback from register outputs Q1 and Q2 gives a HIGH input to D0 when both are HIGH or LOW. Again the design is based on a shift register, and this time the feedback is applied to the least significant register in the chain. The Linear Feedback Shift Register or LFSR shown in Figure 2 is implemented in three CLBs for small state machine designs. CLB placement and routing could be optimized for speed and performance. The LFSR counter would have a "stuck" state if all of the registers were set to a logic HIGH. The feedback input to D0 would remain HIGH and the XNOR inputs from Q1 and Q2 would remain stuck at a logic HIGH. The device has $2^n - 1$ states because in normal counting the "stuck" state cannot be entered. Table 2a shows the sequence and truth table of a three-register, free-running LFSR.

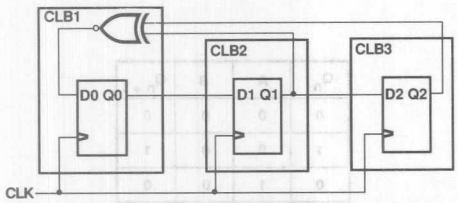


Figure 2. Linear Feedback Shift Register (LFSR)

Modified Linear Feedback Shift Register

The "stuck state" is included in the truth table shown in Figure 2b. If after the count of 3, the state machine could be modified with gating to accommodate the "stuck state" of 7, then an additional state could be used. Shifting a HIGH rather than a LOW into the least significant register would generate state 7, and all of the possible states could be reached. Table 2b shows the new entry in the truth table, and the Karnaugh map in Figure 3b shows the state assignment entry. For example, the map location Q0 = 0, Q1 = Q2 = 1 is the binary code 6. The State Diagram of the counter (Figure 3a), which is derived from the truth table (Table 2b), shows the sequential flow of each state into the next. From information in the State Diagram and State Assignment Map, a State Excitation Map may be developed as shown in Figure 3c. The next state, or excitation state from STATE 6 in Figure 3a is STATE 5. The corresponding entry of 6 in the Assignment Map is replaced by 5 in the Excitation Map and represents the transition from 6 to 5. Figure 3c is developed into Figure 3d by converting the entries to binary notation and placing the condition of the least significant bit into the Karnaugh map shown in Figure 3d. Minimization gives a Boolean equation:

$$Q0 := \sim Q1^*\sim Q2 + Q0^*\sim Q2 + \sim Q0^*Q1^*Q2 \qquad (1)$$

Equation (1) is the functional input to the least significant register in the chain. The other registers in the chain require only the direct inputs from the preceding registers, so no additional minimization is required. Figure 4 shows the final gate implementation which can be incorporated into CLB 1 replacing the XNOR gate.

The advantages of using every possible state in the counter avoids the possibility of a stuck state which, if entered, will prevent the entire design from functioning.

A very important consideration of the LFSR in CLB-intensive designs is the ability to use IOB registers in the counter. While the IOBs have no logic functionality they do have registers. The LFSR is based on the shift register, so if CLB resources run out, IOBs could be used.

COUNT	Q0	Q1	Q2
0	0	0	0
1	1	0	0
3	1	1	0
6	0	1	1
5	1	0	1
2	0	1	0
4	0	0	1

Table 2a. Truth Table of a Linear Feedback Shift Register

COUNT	Q0	Q1	Q2
0	0	0	0
1	1	0	0
3	1	1	0
7	1	1	1
6	0	1	1
5	1	0	1
2	0	1	0
4	0	0	1

Table 2b. Truth Table of a Modified Linear Feedback Shift Register

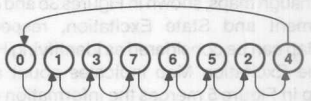


Figure 3a. State Diagram for Modified Linear Feedback Shift Register

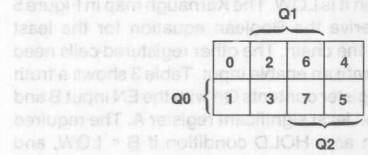


Figure 3b. State Assignment Map

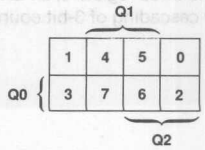
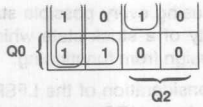


Figure 3c. State Excitation Map



Q0 := ~Q1*~Q2 + Q0*~Q2 + ~Q0*Q1*Q2

Figure 3d.

From the state excitation map, the condition of the least significant register Q0 is entered. Minimization gives the equation required for Q0 IN as a Boolean function of Q0, Q1 and Q2. When implemented, the modified LFSR will count through all states.

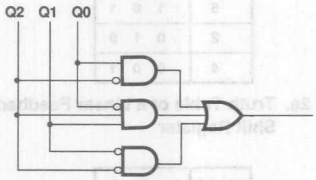
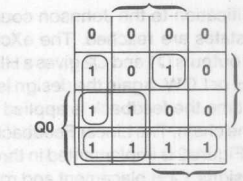


Figure 4.

If this circuit is used to replace the exclusive NOR gate in CLB1 of Figure 2 then the count shown in Table 2b will be realized enabling a count through 2^n states.

Provisions for Deeper Counting

Deeper counters may be made from cascading 3-bit LFSRs. The same three-bit module may be repeated any number of times but the original circuit needs modification. When cascading counters it is essential to use an ENABLE or CARRY input to the next stage. The Karnaugh maps, shown in Figures 3b and 3c, represent State Assignment and State Excitation, respectively. The Assignment Map can be considered as identifying HOLD conditions while the Excitation Map indicates count activity. The Karnaugh map in Figure 5 merges the information contained in both Figures 3b and 3c, and an additional ENable input qualifying COUNT and HOLD activities. The registers will count if EN is HIGH and HOLD when it is LOW. The Karnaugh map in Figure 5 was developed to derive the Boolean equation for the least significant register in the chain. The other registered cells need modifying to incorporate an enable input. Table 3 shows a truth table of the current register contents Qn with the EN input B and the input from the next least significant register A. The required output Q_{n+1} is given as a HOLD condition if B = LOW, and when HIGH, data from the A input will be clocked into the register. For each of the three registers, an EN input will enable count activity to allow cascading of 3-bit counter modules.

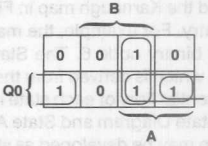



```
Q0 := Q0*~EN          ;HOLD FUNCTION.
    + Q0*~Q2*EN       ;COUNT
    + ~Q1*~Q2*EN      ;COUNT
    + Q2*Q1*~Q0*EN    ;COUNT.
```

Figure 5. Incorporating a Count Enable Input

Q_n	A	B	Q_{n+1}
0	0	0	0
1	0	0	1
0	1	0	0
1	1	0	1
0	0	1	0
1	0	1	0
0	1	1	1
1	1	1	1

Table 3.



General shift register with count enable. A = input from the next least significant register and B = enable count. Q is the state of the internal register. This Karnaugh Map has been developed from Table 3.

Q := Q*~B + Q*A + A*B

Figure 6.

Propagation Carry

To link counter modules together, it is necessary to pass a count enable signal from one module output to the next input. Figure 7 shows the configuration of CLBs 1 to 3 linked to CLB 4, which is configured as a carry generator providing an enable output to the next modified LFSR. The next counter module will not increment until it receives an assertion on its count enable input. Figure 8 shows the Karnaugh map from which the lookahead-carry output was generated. The penultimate count in the sequence, as shown in Figure 2a is two. This count is decoded in the CLB 4 where the registered output is passed to the next counter module and delayed by one clock cycle. The concept of decoding the penultimate count, and delaying it by one clock cycle provides a fast method for propagating the lookahead carry signal. If the ultimate count is decoded by a combinational circuit, the next counter module will have to wait for the carry to propagate through logic gates before the next clock pulse can be applied. Registering the next-to-last count allows premature carry propagation, providing the carry enable to be set up prior to the next clock edge.

Figure 9 shows two identical LFSRs linked by a carry generator's CLBs 5,6 and 7 and a propagate carry configuration in CLB 8. The propagate carry circuit is a combinational circuit that enables the generated carry output, from CLB 4 to the ENB output, as shown in Figure 9. A third set of modified LFSRs may be added if a synchronous nine-bit counter is required.

Applications for a nine-bit counter might include a refresh counter for 256 K Dynamic RAMs. The EN input to CLB 1 can be used to hold off count increment activity. If an LFSR with $2^n - 1$ states were used, then one row in the DRAMs would never get refreshed, so the modified LFSR would be the appropriate choice. Moreover, the count sequence is not important because it does not have to be a binary code, just as long as the DRAMs are refreshed at regular intervals for every row.

Another application might be as a counter in a video controller. The system could count through the required states, and additional gating could be used to generate line and frame sync pulses at certain states during the count sequence.

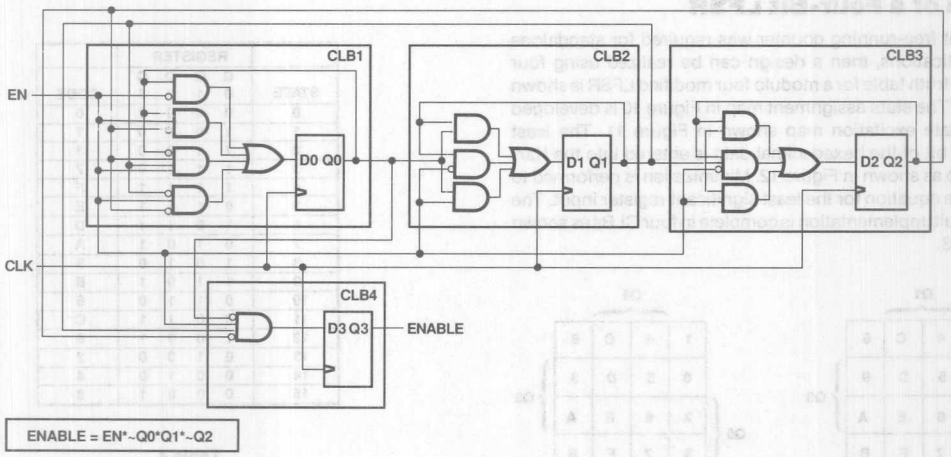


Figure 7.

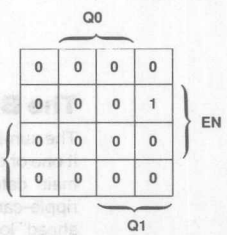


Figure 8. To Enable a Second Stage as a Synchronous Counter a Carry Output Must Be Generated from the First Three Registers. The Enable Output from CLB 4 Is Decoded from the Penultimate Count and Clocked Through the Register. When the Ultimate Count Is Reached the Enable Input to the Next Stage Is Asserted.

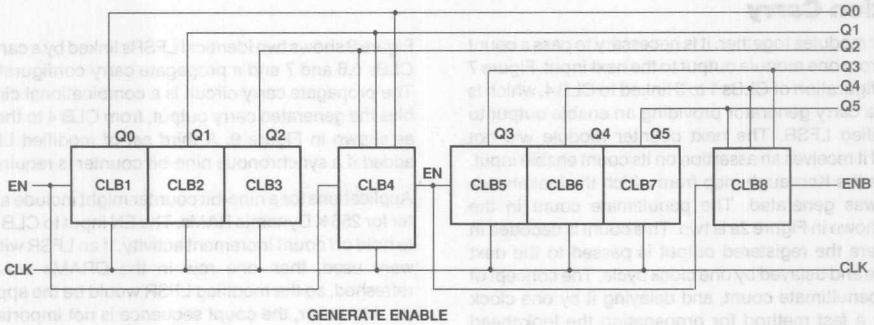


Figure 9. Two LFSRs Are Linked by the Propagate Enable Logic Contained in CLB4. CLB5, 6 and 7 Contain the Same Logic as CLB1, 2 and 3 to Create a Modulo 6 Counter with Sixty-four States. Propagate Enable is Decoded from the Ultimate Count of the Second LFSR Stage. It Can Be Used as the Enable Input to a Third LFSR Stage to Make a 9-Bit Counter.

Design of a Four-Bit LFSR

If a four-bit free-running counter was required for standalone count applications, then a design can be realized using four CLBs. The truth table for a modulo four modified LFSR is shown in Table 4. The state assignment map in Figure 10 is developed into the state excitation map shown in Figure 11. The least significant bit of the hexadecimal data is entered into the Karnaugh map as shown in Figure 12. Minimization is performed to develop the equation for the least significant register input. The actual circuit implementation is complete in four CLBs as shown in Figure 13.

	REGISTER				
STATE	Q 0	Q 1	Q 2	Q 3	CODE
0	0	0	0	0	0
1	1	0	0	0	1
2	1	1	0	0	3
3	1	1	1	0	7
4	1	1	1	1	F
5	0	1	1	1	E
6	1	0	1	1	D
7	0	1	0	1	A
8	1	0	1	0	5
9	1	1	0	1	B
10	0	1	1	0	6
11	0	0	1	1	C
12	1	0	0	1	9
13	0	1	0	0	2
14	0	0	1	0	4
15	0	0	0	1	8

Table 4.

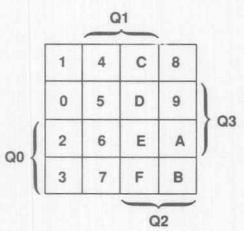


Figure 10. State Assignment Map

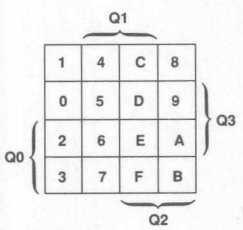


Figure 11. State Excitation Map

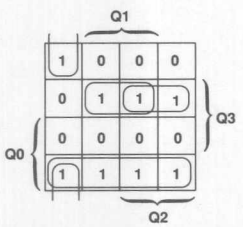


Figure 12. State Excitation Map for the Least Significant Register

Q0 := Q0*~Q3 +~Q0*Q1*Q3 +~Q0*Q2*Q3 +~Q1*~Q2*~Q3

The Binary Counter

The number of states available in a binary counter is 2^n, making it one of the most register-efficient types of counters. The three main categories of binary or weighted counters are ripple, ripple-carry, and lookahead-carry. As the amount of "look-ahead" logic decoded at each counter stage increases, the performance of the binary counter also increases. As a result, lookahead-carry counters are the fastest of the three categories of binary counters. On the other hand, the more "look-ahead" logic there is, the more decoding is needed. Increased decoding requires extra logic and routing resources which may be disadvantageous if these resources become scarce.

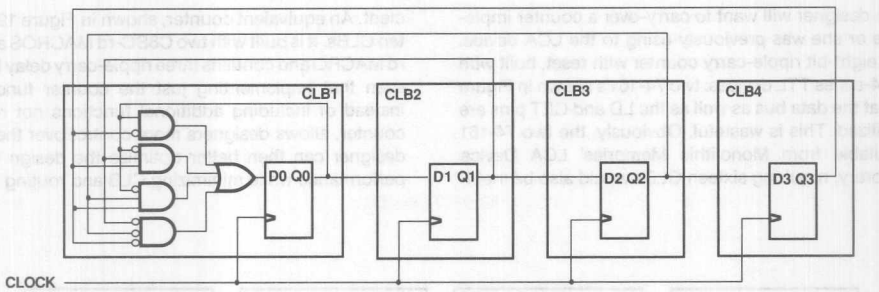


Figure 13. Modified LFSR Modulo 4 Counter

The Ripple Counter

Ripple counters are asynchronous in nature and do not generate carry signals. When the output of each counter stage clocks the next stage on the negative clock transition, a ripple effect is induced (thus the name). A schematic representation of a six-bit ripple counter is shown in Figure 14. One of the benefits of a ripple counter is that it requires few resources. Only one CLB per counter bit is needed to implement the counter and routing is simple regardless of the counter length. The tradeoff for this simplicity, however, is that ripple counters cannot be modified to be loadable or up/down which restricts their operation to be free-running, and they have lower performance.

The overall performance of a ripple counter degrades with each counter bit by one CLB delay time. This can be shown by the equation below:

$$\text{Ripple Counter} = N * \frac{(\text{Clock to Output Delay})}{\text{Clock Period}} \quad (2)$$

where N = the number of ripple counter flip-flops

The overall counter clock period must therefore be greater than or equal to the total cumulative CLB delay.

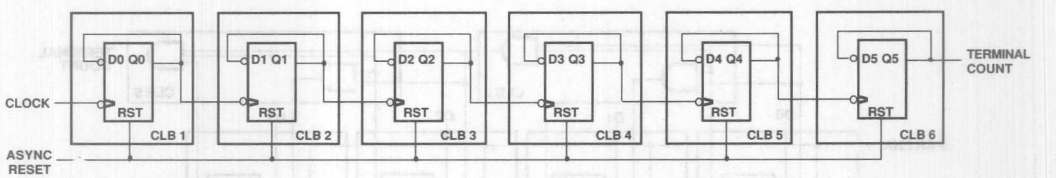


Figure 14. Schematic of a 6-Bit Binary Ripple Counter. Ripple Counters Are Easy to Design and Are Cascadable to Nearly Any Length. They Are, However, Asynchronous and Are Not Recommended for Most Designs.

The Ripple-Carry Counter

The ripple-carry counter is similar to the ripple counter. The ripple-carry counter has carry signals, however, whereas the standard ripple counter does not. Each carry signal propagates to the next counter stage to produce the counting sequence. This cascaded connection is called ripple-carry.

The ripple-carry counter differs from the ripple counter in the respect that it is synchronous in operation. Because of this, the ripple-carry counter provides more reliable operation and better performance. Performance still degrades with each additional counter stage though, due to the inclusion of combinational logic from the previous counter stage. Like the ripple counter, the ripple-carry counter requires only one CLB per counter bit and is easily cascadeable.

One example of a four-bit binary ripple-carry counter is shown in Figure 15. Here, a modulo-16 counter with count enable (CE) and reset is built with four CLBs and three levels of ripple logic. The same counter could be designed with two levels of ripple logic, if the two-input AND gate in CLB2 becomes a three-input AND gate, and the signals CE and Q0 from CLB1 were carried over to CLB2 and ANDed with Q1.

Another four-bit counter, this time with parallel enable (load), count enable, and reset, is shown in Figure 16. Here, just two ripple levels of logic were designed in, using two C8BCP MACROS from Monolithic Memories' Macrocell Library. Six CLBs, two more than the previous four-bit counter, were used in this design because of the addition of parallel enable circuitry and the reduction of ripple-carry logic levels. The Karnaugh map corresponding to the registered CLBs is shown in Figure 17.

Sometimes a designer will want to carry-over a counter implementation he or she was previously using to the LCA device. Consider an eight-bit ripple-carry counter with reset, built with two typical 74-series TTL devices: two 74-161's shown in Figure 18. Notice that the data bus as well as the LD and CET pins are not being utilized. This is wasteful. Obviously, the two 74-161 macros, available from Monolithic Memories' LCA Device Macrocell Library, requiring sixteen CLBs would also be inefficient. An equivalent counter, shown in Figure 19, requires only ten CLBs. It is built with two C8BC-rd MACROS and one C4BC-rd MACRO and contains three ripple-carry delay levels. It can be seen that implementing just the counter functions desired, instead of including additional functions not required in the counter, allows designers more control over their design. The designer can then better optimize the design for speed and performance while minimizing CLB and routing resources.

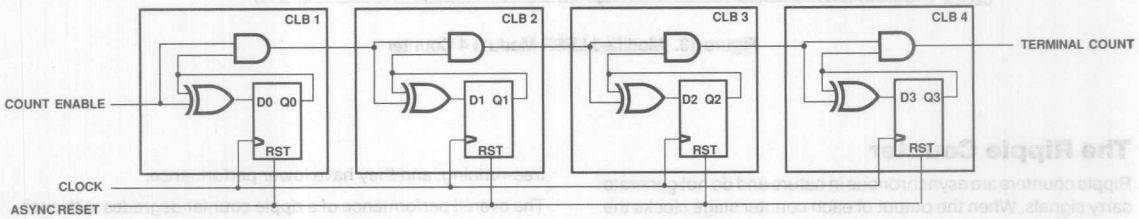


Figure 15. Schematic of a 4-Bit Ripple-Carry Counter. Although the Ripple-Carry Counter Is Synchronous, the Carry Input to the Next Stage Is Conveyed Through Combinational Logic. The Propagation Delay Due to this Logic Must be Taken into Account.

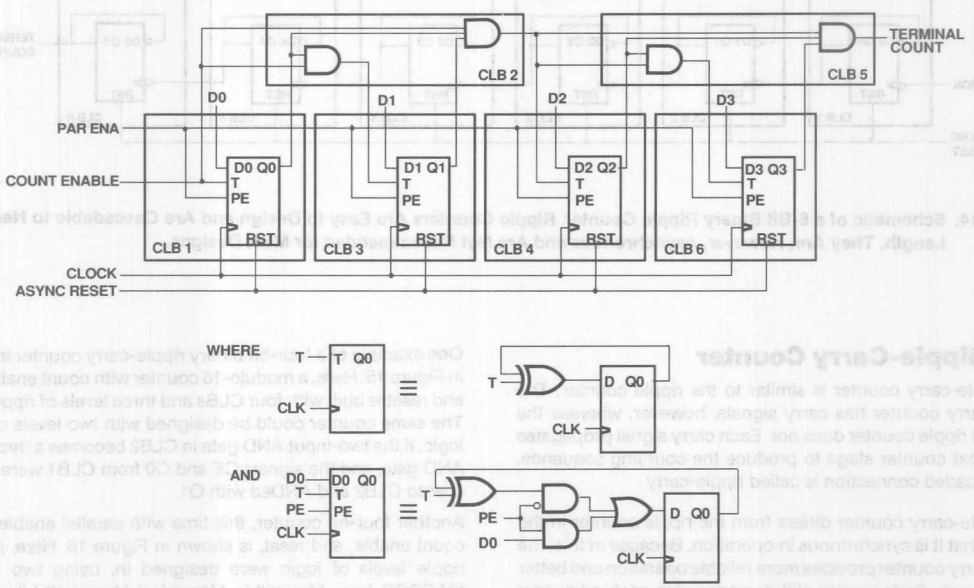


Figure 16. Schematic of a 4-Bit Ripple-Carry Counter. With Parallel Enable, Count Enable, and Reset, this Counter Only Contains Two Levels of Ripple-Carry Logic.

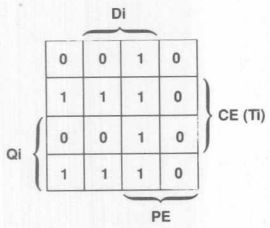


Figure 17. Karnaugh Map for the Registered CLBS in Figure 16.

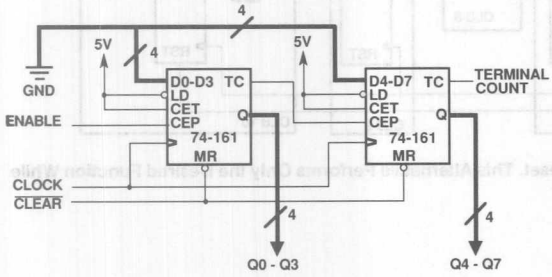


Figure 18. Schematic of an 8-Bit Counter with Reset Using Two 74-161 TTL Devices. Since All of the Available Logic Is Not Utilized, the Counter Should Be Designed More Efficiently for Use in the LCA Device.

The Lookahead-Carry Counter

When the performance of binary ripple and ripple-carry counters is not adequate, synchronous binary lookahead-carry counters can be a solution. A lookahead-carry counter incorporates all the previous counter outputs into a single lookahead-carry signal for each counter stage. This logic reduction minimizes the overall delays of the design which makes the performance of lookahead-carry counters the highest among binary counters.

With lookahead-carry counters, the complexity of the design increases with each counter bit as the decoding inputs become ever wider. As a result, these counters usually need more CLBs and routing resources to implement than other binary counters. The equations which characterize an n-bit lookahead-carry counter with count enable and parallel enable are listed below:

Q0 := ((/PARENA * CE) @ Q0) + (PARENA * DO)

Q1 := ((/PARENA * CE * Q0) @ Q1) + (PARENA * D1)

Q2 := ((/PARENA * CE * Q1) @ Q2) + (PARENA * D2) . . .

and

Qn := ((/PARENA * CE * Q1 * ... * Qn-1) @ Qn)
+ (PARENA * Dn)

To decrease the complexity of the counter, one or more of the control signals can be removed.

For a ten-bit lookahead-carry counter with reset, a minimum of fourteen CLBs are needed as shown in Figure 20. The logic for the counter was partitioned as follows:

CLB 1	Q0 := CE @ Q0
CLB 2	Q1 := (CE * Q0) @ Q1
CLB 3	Q2 := (CE * Q0 * Q1) @ Q2
CLB 4	Q02 = Q0 * Q1 * Q2 * CE
CLB 5	Q3 := Q02 @ Q3
CLB 6	Q4 := (Q02 * Q3) @ Q4
CLB 7	Q5 := (Q02 * Q3 * Q4) @ Q5
CLB 8	Q35 = Q3 * Q4 * Q5
CLB 9	Q6 := (Q02 * Q35) @ Q6
CLB 10	Q36 = Q3 * Q4 * Q5 * Q6
CLB 11	Q7 := (Q02 * Q36) @ Q7
CLB 12	Q8 := (Q02 * Q36 * Q7) @ Q8
CLB 13	Q38 = (Q36 * Q7 * Q8)
CLB 14	Q9 := (Q02 * Q38) @ Q9

The first seven bits of the counter were decoded in a similar fashion to the counter in Figure 19. The next three bits continue to minimize the amount of ripple-carry logic implemented to make it a lookahead-carry counter.

When routing a design such as this in the LCA device, it is best to place the CLBs lengthwise. Then, if the high fan-out output signals for Q02 and Q36 CLBs can be routed through "long line" interconnects, the routing-dependent delays can be reduced allowing the counter to perform at its maximum potential speed.

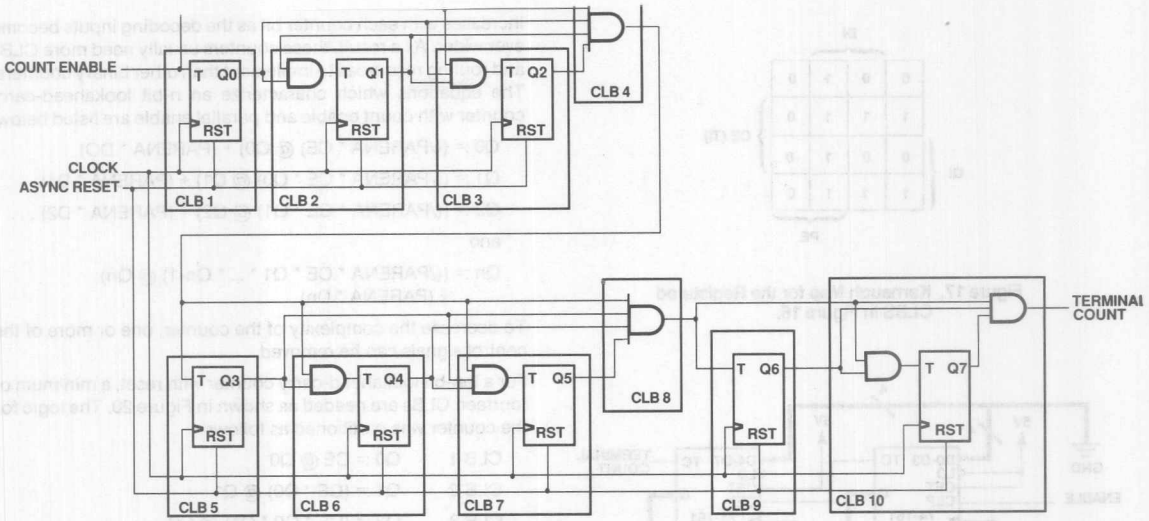


Figure 19. An Efficient Implementation of an 8-Bit Counter with Reset. This Alternative Performs Only the Desired Function While Maximizing CLB and Routing Utilization.

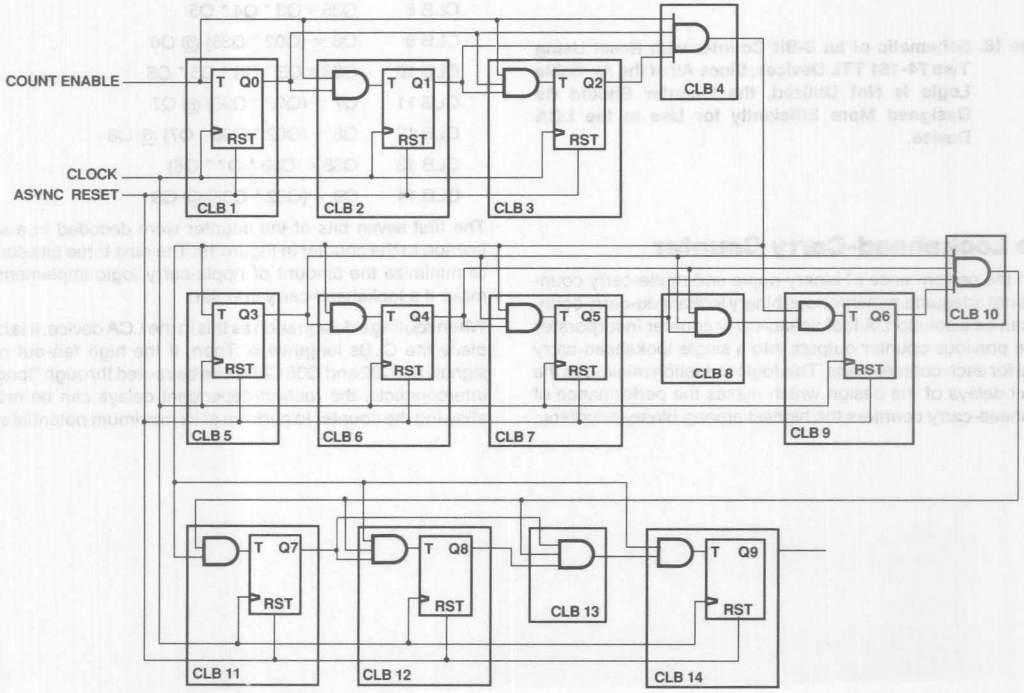


Figure 20. Schematic of a Synchronous Counter with Reset as in Figure 19, but with Lookahead-Carry Logic. As More Counter Bits Are Added, the Design Becomes Increasingly Complex Due to Wider Gating.

The Up/Down Counter

The up/down counter can be a very useful device. For instance, status counters or address counters employed in DMA systems, dual slope integrators, and delta modulation systems usually use up/down counters. In operation, an up/down counter will count UP when all previous counter bits are HIGH and will count DOWN when all previous counter bits are LOW. Thus, in an up/down counter, each register output in a CLB will toggle when:

- Q0 through Qn-1 are HIGH and the counter direction is UP, or
- Q0 through Qn-1 are LOW and the counter direction is DOWN.

This translates to:

$$Qn := Qn \ @ \ ((Qn\text{-}1 * Qn\text{-}2 * \ldots * Q1 * Q0 * UP) + (/Qn\text{-}1 * /Qn\text{-}2 * \ldots * /Q1 * /Q0 * /UP)). \quad (3)$$

Design of the counter can become more complicated by adding new features to it such as count enable, reset, or load (parallel enable). Because of this, it is best to limit the number of control signals which would require more complex logic. For example, a counter which only needs to be reset during initialization will not need additional reset capability since all registers are reset upon initialization of the LCA device.

Simple ripple-carry binary up/down counters require 2 * N - 4 CLBs for implementation where N is the number of counter bits. For example, eight CLBs would be needed for a six-bit up/down counter and twenty-eight CLBs would be needed for a 16-bit up/down counter. An example of a six-bit ripple-carry up/down counter configured in CLBs is given in Figure 21.

The general equations used for this ripple-carry counter design were derived from equation 3 and are as follows:

Running Total of CLBs	Equations
1	Q0 := /Q0 C2X = UP * Q0 * Q1
2	Q1 := Q1 @ ((UP * Q0) + (/UP * /Q0)) C2Y = /UP * /Q0 * /Q1
3	Q2 := Q2 @ (C2X + C2Y)
4	Q3 := Q3 @ ((Q2 * C2X) + (/Q2 * C2Y))
5	C3X = C2X * Q2 C3Y = C2Y * /Q2
6	Q4 := Q4 @ ((Q3 * C3X) + (/Q3 * C3Y))
7	C4X = C3X * Q3 C4Y = C3Y * /Q3
8	Q5 := Q5 @ ((Q4 * C4X) + (/Q4 * C4Y))

To build a simple ripple-carry 16-bit up/down counter, this algorithm would be continued:

$$Qn := Qn \ @ ((Qn\text{-}1 * Cn\text{-}1X) + (/Qn\text{-}1 * Cn\text{-}1Y))$$

where:

Cn-1X = Cn-2X * Qn-2
Cn-1Y = Cn-2Y * /Qn-2
Cn-1X, Cn-1Y are > or = C2X, C2Y

and

C2X = UP * Q0 * Q1
C2Y = /UP * /Q0 * /Q1.

The test file that was used to verify the logic of the six-bit up/down counter in Figure 21 is shown in Figure 22. It was generated by running the SIMGEN program included with the XACT Development System. This file, with a .DAT extension, was modified to include the desired input test vectors which would be executed. Once in the P-Silos™ simulator, "IN <filename>.DAT" is entered. This command automatically inputs the netlist to the simulator. Finally, a time period is entered which informs the simulator of the length of time the simulation should run. For instance, "SIM 0 10000" could be entered. The outputs will be plotted in a table format and will list the signals specified.

An alternative method for designing up/down counters is to use lookahead-carry logic, based on equation 3. This performance-driven method, however, is quite CLB intensive due to the wide gating of input signals required for implementation. Thus, for a six-bit lookahead-carry up/down counter, sixteen CLBs would be needed, whereas for the six-bit ripple-carry up/down counter, ten CLBs were needed. Therefore, it is advantageous to use the lookahead-carry method when maximum performance is needed.

A design of an n-bit ripple-carry up/down counter that can synchronously RESET and LOAD new values into the counter also becomes more difficult. From the general expression below, derived from state tables and Karnaugh maps, any up/down counter can be designed.

$$Qn = (/RESET * LOAD * Dn) \ @ ((/RESET * /LOAD * UP * Qn\text{-}1 * \ldots * Q1 * Q0) + (/RESET * /LOAD * /UP * /Qn\text{-}1 * \ldots * /Q1 * /Q0))$$

where RESET and LOAD are active HIGH.

There are many ways to design up/down counters. The look-ahead carry method, though it consumes a great deal of resources, usually enhances performance. The best way to design the up/down counter for systems that do not require high speed, is to use the ripple-carry method. This method is usually the most straightforward and requires the least number of resources.

5

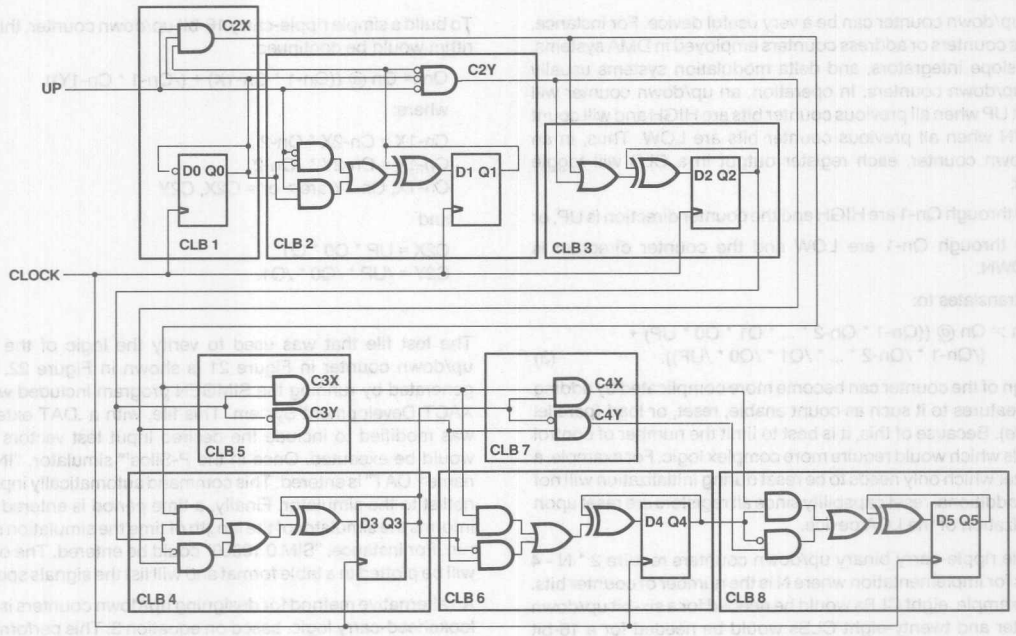


Figure 21. Schematic of a 6-Bit Up/Down Counter. Only Eight CLBs are Required for Implementation.

```
$
$ Simulation file for design 'FIG21.LCA' type '2064NL68-70'
$ Created by XACT Ver. 1.30 at 14:14:31 JUL 21, 1987
$
!INPUT FIG21.sim

$ INPUTS:
GLOBALRESET- .CLK 0 S0 1 S1 $ Initial pulse to reset latches
CLK          .CLK 0 S0 1000 S1 2000 S0 .REP 0
UP           .CLK 0 S0 128000 S1 256000 S0 270000 S1 280000 S0

  .MONITOR CLK ; UP ; Q0 Q1 Q2 Q3 Q4 Q5
  .TABLE   CLK ; UP ; Q0 Q1 Q2 Q3 Q4 Q5
```

Figure 22. Input File to P-Silos

Summary

Many ways exist for implementing counters in the LCA device. The ideal counter design for a specific application depends on the desired performance and the available logic and interconnection resources. See Table 5. For example, three main types of counters, Johnson, Linear Feedback Shift Register, and binary counters were discussed. Each of these counters utilizes different features of the LCA device but each can be optimized to perform a required function. Moreover, it was seen that by implementing only the logic necessary for a required function, resource efficiency was increased and by implementing more lookahead-carry logic or using LFSRs, performance was increased.

COUNTING METHOD	COUNTER TYPE	MODULO	CLB EFFICIENCY	ROUTING EFFICIENCY	PERFORMANCE	ADVANTAGES/ DISADVANTAGES
Non-binary	Johnson	2n	Poor	Excellent	Excellent with low modulo	High performance, easy routing Glitch-free decoding Low-register efficiency
	Linear feedback shift register or modified LFSR	2^n - 1 or 2^n	Good	Very good	Very good decreases with increasing modulo	Good performance at high modulos
Binary	Ripple	2^n	Excellent	Excellent	Poor	Low resources, easy routing but slow and asynchronous
	Ripple-carry	2^n	Very good	Good	Good	Good general binary counters
	Lookahead carry	2^n	Good but decreases with increasing modulo	Good but decreases with increasing modulo	Good for high-performance binary counters	Highest performance binary counter requires more CLB and routing resources

Table 5. Summary of Counter Types and Their Characteristics

COUNTING METHOD	COUNTER TYPE	MODULO	CLB EFFICIENCY	ROUTING EFFICIENCY	PERFORMANCE	ADVANTAGES/ DISADVANTAGES
Non-binary	Johnson	2n	Poor	Excellent	Excellent with low modulo	High performance, easy routing Glitch-free decoding Low-register efficiency
	Linear feedback shift register or modified LFSR	$2^n - 1$ or 2^n	Good	Very good	Very good decreases with increasing modulo	Good performance at high modulos
Binary	Ripple	2^n	Excellent	Excellent	Poor	Low resources, easy routing but slow and asynchronous
	Ripple-carry	2^n	Very good	Good	Good	Good general binary counters
	Look-ahead carry	2^n	Good but decreases with increasing modulo	Good but decreases with increasing modulo	Good for high-performance binary counters	Highest performance binary counter requires more CLB and routing resources

Table 5. Summary of Counter Types and Their Characteristics

Time Division Multiplexing with the LCA Device

C.B. Lee and Theresa Shafer

Abstract

High-speed data communication lines are efficiently utilized when low-speed signals are time division multiplexed onto high-speed lines. Time division multiplexing requires data buffering, rate adaption and data selection. The data selection fits into a single CMOS Logic Cell™ Array (LCA device). The LCA device is designed and optimized using the FutureNet® schematic capture package, Monolithic Memories' automatic place and route software, and the XACT™ Design Editor System.

5

2175 Mission College Blvd. Santa Clara, CA 95054-1592 Tel: (408) 970-9700 TWX: 910-338-2376 TWX: 910-338-2374
Monolithic Memories

Time Division Multiplexing with the LCA Device

C.B. Lee and Theresa Shafer

Introduction

The Logic Cell Array (LCA device) implements a multiplexer and counter used in time division multiplexing. With the device's flexible I/O pins, the multiplexer and counter are implemented into a single CMOS device. This application note covers time division multiplexing and the design of a multiplexer and counter. Additional information on how to design with an LCA device can be found in the LCA design methodology chapter in Monolithic Memories' LCA Design and Applications Handbook. For programming the LCA device, refer to "Configuring the LCA Device", Monolithic Memories' Application Note 182.

Principles of Multiplexing

Multiplexing efficiently utilizes data communication lines by combining multiple low-speed signals into a single, high-speed line. The two methods of performing multiplexing are Time Division Multiplexing (TDM) and Frequency Division Multiplexing (FDM). TDM divides the transmission bandwidth into equal time slots where each input signal is assigned one time slot per time cycle. Once assigned, that time slot is not used by any other input. Figure 1 shows a multiplexer combining three low-speed signals into one high-speed line. Terminal A transmits during the first time slot, B transmits during the second time slot, and C transmits during the third time slot. This sequence is repeated every time cycle. FDM, on the other hand, divides the frequency spectrum among logical channels where each channel has full bandwidth of its assigned frequencies. Analog systems usually use FDM.

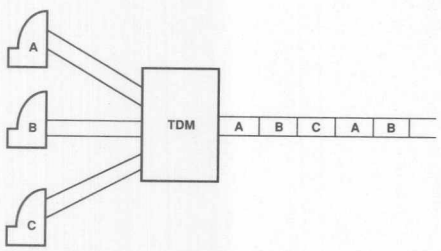


Figure 1. Time Division Multiplexing

There are many classes of TDMs including bit interleave, character interleave, statistical TDM, and T1 multiplexers. In bit interleaved TDM, each input is assigned to one time slot. Each time slot's length is one bit. Character interleaved TDM is similar to bit interleave, except that each time slot represents one character. Statistical multiplexing assigns the bandwidth into unequal slots where only the active incoming lines are assigned slots. The slots are of variable length, so each input is assigned the amount of bandwidth needed. The T1 standard specifies twenty-four 64 kbps channels multiplexed on a 1.544 Mbps high-speed link.

Data Selection/Time Slot Assignment

To transmit data from terminal A in the first time slot as shown in Figure 1, data buffering, rate adaption and data selection must be performed. Data buffering and rate adaption are required since the rates of the lines to be multiplexed are slower than the high-speed link. A serializing FIFO or shift register can implement the required buffering. Rate adaption is performed by a clock/shift scheme tailored to the exact application. Once buffered, the data must be selected in the proper order. Several methods can be used for data selection; one method is a 32-to-1 multiplexer. To implement sequential selection of input lines, an on-board counter selects the multiplexer's output. However, for random selection needed in statistical multiplexing, more flexibility is needed. A 2-to-1 multiplexer provides this flexibility by selecting between the counter and random selection inputs.

Comparison of Design Methodologies

The 32-to-1 multiplexer and counter design can be implemented in several ways. Briefly, we will evaluate discrete logic, PAL and LCA device implementations.

For this design, thirty-eight inputs, five registers, and one output are required. Conventional PLD devices do not meet this large I/O requirement. A single PLD device is not suited for this specific application because flexible I/O and buried registers are not supported. Instead, the design must be divided into four sections. The counter fits in a PALC16R6Z device and the 32-to-1 multiplexer requires three PALC20L8Z devices.

Since a large 32-to-1 multiplexer is not available as a discrete part, it must be built using smaller multiplexers. The total chip count, using discrete logic, is five devices: two 74AS850s, two 74ALS257s and one 74AS867. A combination of PAL devices and discrete logic devices is possible, however, it is also a multiple chip solution.

Monolithic Memories' LCA device is a high-density programmable CMOS circuit with flexible I/Os. It has forty pins which are user-defined as either inputs, outputs, or bidirectional. The LCA device meets the I/O requirement for this design. Since the LCA circuit allows multiple logic levels, implementing the counter does not use up output pins. Moreover an LCA device enables the 32-to-1 multiplexer with counter to be implemented in a single low-power device.

Detailed LCA Design

The M2064 LCA device in a 48-pin DIP is used in this design, although both PLCC and PGA packages are available. Eight of the forty-eight pins are reserved for programming, power, and ground, and the remaining forty I/O pins are available to the user. Table 1 shows the efficient use of the package pinout.

PIN	DESCRIPTION	TOTAL PINS
ADDR(4:0)	External Address Inputs	5 inputs
CK	External Clock Input	1 input
LD	Counter Load Input	1 input
D(31:0)	32-to-1 Multiplexer Data Inputs.	32 inputs
OUT	Output	1 output
TOTAL I/O		40 pins

Table 1. 32-to-1 Multiplexer Pins

Figure 2 shows the block diagram of the 32-to-1 multiplexer and the counter. The select lines of the 32-to-1 multiplexer are either the address lines or the output of the counter. When LD is deasserted, the 5-bit counter sequentially selects each input. Asserting the LD signal loads the counter with the external addresses.It also selects the external address for the 32-to-1 multiplexer's select lines. This permits random input selection from the external address lines which supports statistical multiplexing.

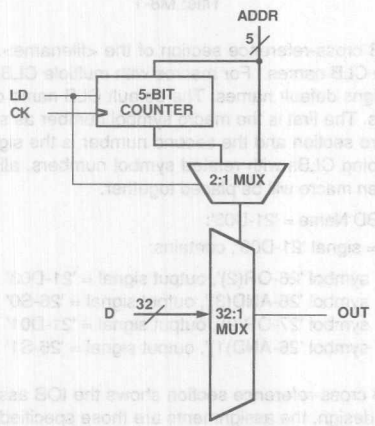


Figure 2. Block Diagram

The select signals for the 32-to-1 multiplexer are determined by the LD signal. When LD = 0, the counter selects the multiplexer's output. When LD = 1, the counter is loaded and the external address selects the output. Also, the LD asynchronously loads the 5-bit counter. Table 2 summarizes the device operation.

INPUTS	OUTPUT	FUNCTION
LD ADDR(4:0)	OUT	
0 XXXXX	D(CNT)	Counter as Select
1 ADDR(4:0)	D(ADDR)	Address as Select and Load Counter

Table 2. 32-to-1 Multiplexer Function Table

The design was entered with FutureNet's DASH schematic capture package using high-level macros. Then the schematic capture file was translated into an LCA design file. The translation software, called PIN2LCA, partitions the design into CLBs and generates the equations for the CLBs' configuration. The translation software produces an unrouted LCA design file. This design file requires final placement of CLBs and interconnect routing which was performed using Monolithic Memories' APR software, an automatic place and route program. After the design is placed and routed, it was optimized with Monolithic Memories' XACT Design Editor System. The XACT system provides a graphic interface to manually optimize the CLB logic functions and connections.

The logic functions and interconnects of an LCA device are established with CMOS memory cells, so the array is never physically altered, although it is physically programmed. A programmable LCA device can be reconfigured for the prototype. In addition, it can be reprogrammed any number of times in the target system. With reconfigurable devices such as this one, the array can also be programmed on power-up, or whenever the design details need to be changed. For volume production, a hard-array version will be available.

Entering the Design with Futurenet

The schematic entered in FutureNet's DASH is comprised of different components called from the macro library supplied by Monolithic Memories. The M8-1 and M4-1 macros are the multiplexers which implement the 32-to-1 multiplexer as shown in Figure 3. The 5-bit counter was built from INV, C16BPRD, XOR2, and FDM macros. Since the C16BPRD macro is only a 4-bit parallel-load binary counter, the XOR2 and FDM macros were added to form another bit which increased the length to five bits.The GMUX macro, a 2-to-1 multiplexer, selects between the counter and the external address lines.

To translate from FutureNet to LCA, the PIN2LCA program is invoked. This program generates an unrouted <filename>.LCA from the FutereNet's <filename>.PIN. The PIN2LCA software partitions the design into CLBs and generates each CLB's configuration.

The PIN2LCA performs logic reduction by eliminating unused logic to maximize CLB utilization. With this design, for example, C16BPRD's reset logic is not used, and PIN2LCA eliminates all the reset logic in the C16BPRD macro. Another means to maximize CLB utilization is by combining macro logic. At times, this combined logic results in a single CLB which implements logic from two different macros. For designs not utilizing all the CLBs, this logic reduction is not required and may actually hinder placement and delay tradeoffs. If it does produce delay problems, it is possible to edit the CLBs in XACT.

At this stage in the design flow, the PIN2LCA software can generate a P-SILOS simulation file. Since the <filename>.SIM was generated from an unrouted LCA design file, the simulation only has unit delays and does not have actual routing delays. The simulation verifies the logic, but does not give any timing information. It may be necessary to make changes in DASH™ depending on the simulation results.

Place and Route Using APR

Once the logic is finalized, the LCA design must be placed and routed. Monolithic Memories' APR software performs the final placement of CLBs and interconnect routing. The input to APR is an unrouted <filename>.LCA, and a routed LCA design file and report document are the outputs.

At first glance, the APR program seems cumbersome to apply to every application. However, it can be used to create an efficient design. While the software uses random placement, it does allow tailoring for specific requirements.

Initially, the multiplexer and counter design did not route completely. To achieve good placement and 100% routing completion, several things were tried. Some worked well while others did not. An overview of how 100% routing completion was achieved is explained below.

Since this design utilizes only 53% of the LCA device, delay is a larger concern than CLB utilization. In this case, placement is critical to achieve an efficient design with the desired delays. The ideal placement would be for the 5-bit counter and 2-to-1 multiplexer to be placed near the ADDR(4:0), CK, and LD signals. The four M8-1 macros should be near the 8-bit cluster of inputs. The remaining M4-1 macro should be placed near the OUT signal.

In FutureNet, the I/Os are assigned pin numbers so that the signals are clustered into groups. These I/O assignments restrict the APR's placement of the signals and are found in the <filename>.SCP. Other restrictions and options are specified when invoking the APR software. The options, "-a", "-g", "-e", and "-k", tailor the placement for this specific design.

The APR's "-a" option specifies the depth of CLB logic levels which are considered connected. In this design, "-a2" was selected because the C16BPRD and M8-1 macros are implemented in two CLB logic levels.

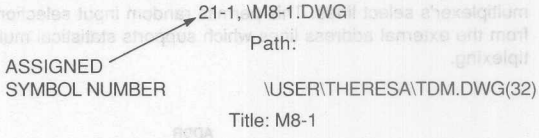
The "-g" and "-e" options are related. The "-g" option specifies the number of CLBs that should be grouped together. The "-e" option determines the number of CLBs in a group which should be evaluated for all possible placements. Since the C16BPRD and M8-1 macros contain four to six CLBs, the options, "-e4" and "-g4", were used. This particular value was selected because it was the largest possible value which maintained the e>=g relationship. Avoiding values where e<g insures as much as possible that the CLBs within each group are placed in the best possible arrangement. This maintains the groups' integrity.

The "-k" option specifies the number of shapes per group. By trial and error, "-k6" works well for this design. With "-k6", six of the best arrangements or shapes were saved for each group of CLBs. Every combination of shaped group is tried with all six other grouped shapes and evaluated for best placement. Generally, the larger values result in better placements, however, run-time grows exponentially. While larger values of "-k" were tried, they did not improve the placement significantly.

Running APR with these options and restrictions did not result in a 100% routed design. Therefore, more application-specific information was required. Noting that the APR software was not fully utilizing the high-level macro information, placement could still be optimized. The APR's report file shows that the CLBs which comprise C16BPRD or the M8-1 macros were not grouped together. Instead, the CLBs were randomly spread over the device. In a constraint file, <filename>.CST, CLBs from the high-level macros could be grouped together (see Appendix A). If necessary, the exact CLB placement could be specified in the same file.

To generate the constraint file, CLBs which implement each high-level macro were identified using the cross-reference file generated by PIN2LCA, <filename>.CRF. The <filename>.CRF file is organized into three cross-reference sections: macros, CLBs, and IOBs.

The macro cross-reference section identifies and assigns a hierarchical symbol number to each macro. The symbol number is used to generate the default CLB names.



The CLB cross-reference section of the <filename>.CRF contains the CLB names. For macros with multiple CLBs, the system assigns default names. The default CLB name consists of two parts. The first is the macro symbol number as specified in the macro section and the second number is the signal name. By grouping CLBs with related symbol numbers, all the CLBs for a given macro will be placed together.

```
CLB BD Name = '21-D03':
    F = signal '21-D03', contains:
        symbol '26-OR(2)', output signal = '21-D03'
        symbol '26-AND(3)', output signal = '26-S0'
        symbol '27-OR(2)', output signal = '21-D01'
        symbol '26-AND)1)', output signal = '26-S1'
```

The IOB cross-reference section shows the IOB assignments. For this design, the assignments are those specified in the FutureNet's schematic.

```
IOB P30: Name = 'D13', Symbol = 'PIN(40)'
    I = signal 'D13'
```

Optimizing the Design with Monolithic Memories' XACT Design Editor System

Monolithic Memories' XACT Design Editor System can increase resource utilization and performance by manually modifying the placement and routing. The XACT system provides a graphic interface to specify the CLB design. All CLB logic functions and connections can be optimized for the designs' needs. With XACT, the designer can partition the design and optimize the placement of logic blocks to gain the utilization and performance required.

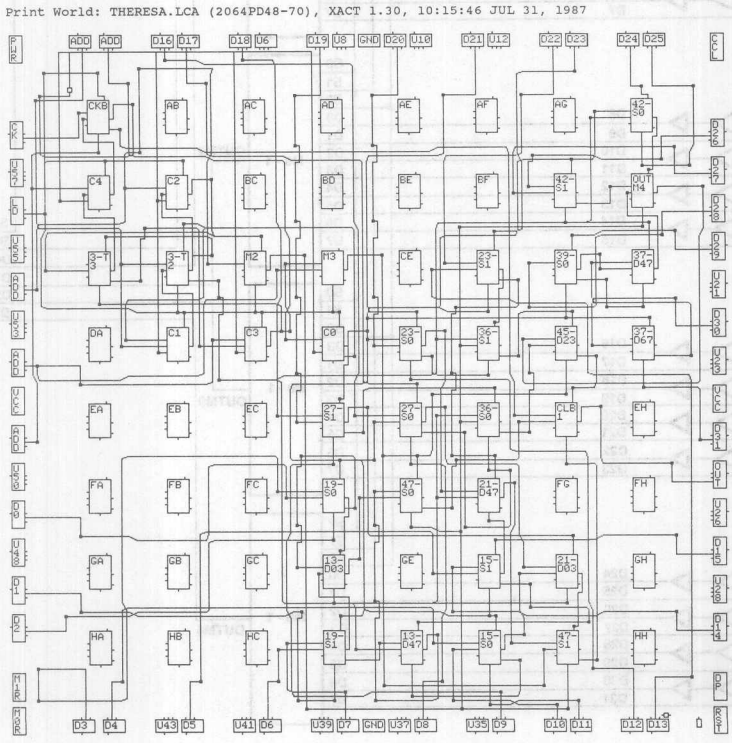
The clock line routing was optimized to reduce clock skew. Using the clock buffer resources, the common clock is driven by the global clock buffer. To optimize internal routing, "long lines" were used for signals with large fanouts such as the select lines for the multiplexers and the LD signal. Use of the "long lines" minimizes the delay for these control signals. The "long lines" were selected by manually routing the signal net through the programmable interconnect points (PIPs).

On the device itself, implementing the 5-bit counter and 2-to-1 multiplexers requires ten CLBs; and the 32-to-1 multiplexer requires twenty-four CLBs, giving a total of thirty-four CLBs. This design utilizes 53% of the LCA device.

Summary and More Information

For time division multiplexing systems, data selection can be implemented in a single CMOS device. FutureNet's DASH schematic capture with Monolithic Memories' LCA macro library was used to implement a design in an LCA device. Placement of CLBs and signal routing were performed by Monolithic Memories' APR software. Optimization, which may be required to improve performance, can be done using Monolithic Memories' XACT Design Editor System. This is useful for placing and routing critical signal nets such as clock or control signals. More information can be found in P-SILOS, FutureNet, and LCA user guides and handbooks.

This design, developed with an LCA device, is available upon request. The FutureNet drawing, LCA design and bit pattern files can be provided for programming the LCA device in an EPROM. Please ask for design XDES07.LCA.



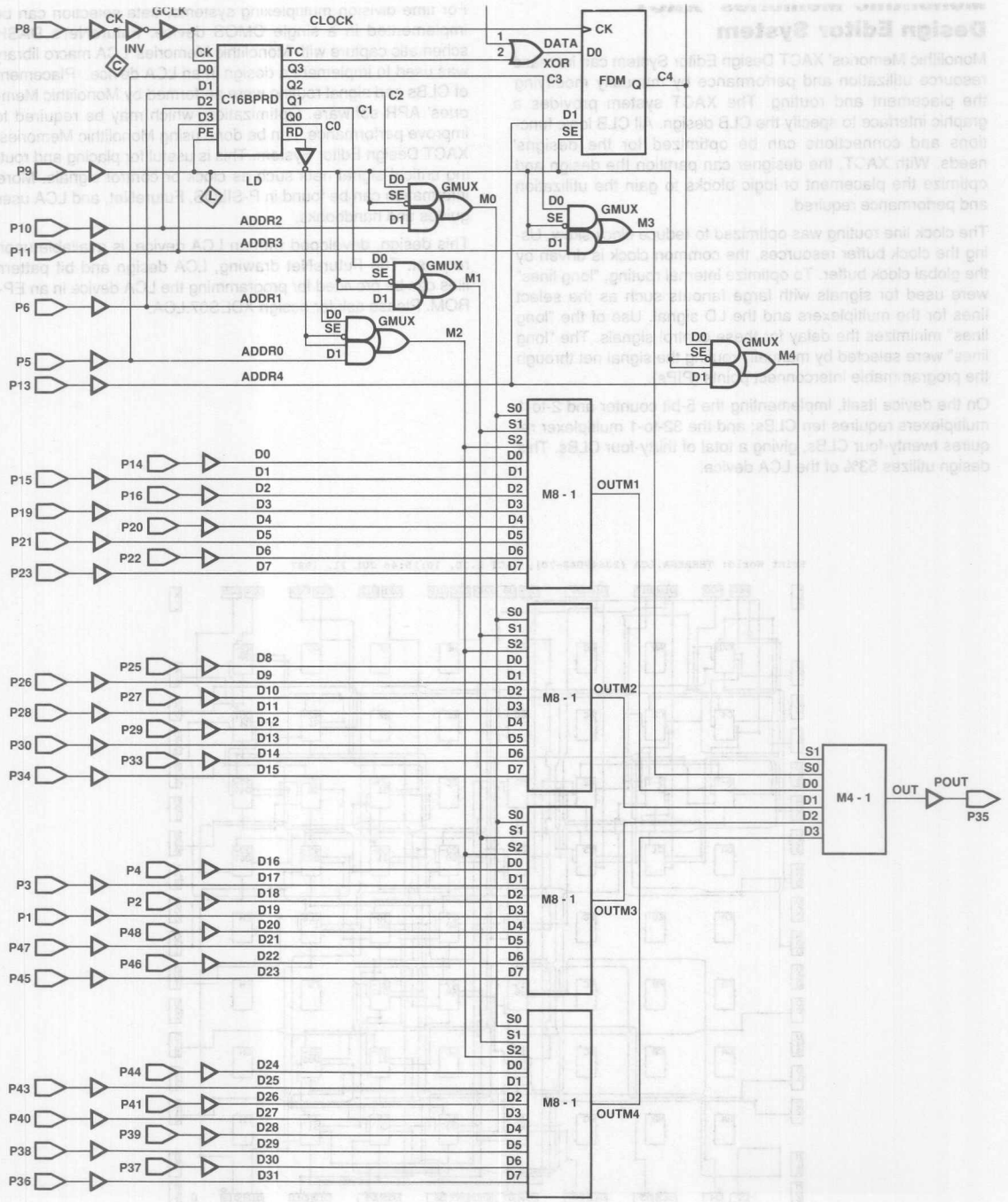


Figure 3. FutureNet Schematic

Appendix A> Constraint File

```
; CONSTRAINT FILE GROUPING MACROS IN BLOCKS

DEFINE GROUP CNT_GROUP
        3-T3 3-T2 CKB
        C4 C2 C3 C1 C0
        M3 M2;

DEFINE GROUP OUT_GROUP
        CLB1 45-D23;

DEFINE GROUP M1_GROUP
        47-S0
        19-S0 19-S1
        13-D03 13-D47;

DEFINE GROUP M2_GROUP
        47-S1
        15-S0 15-S1
        21-D03 21-D47;

DEFINE GROUP M3_GROUP
        27-S0 27-S1
        23-S0 23-S1
        36-S0 36-S1;

DEFINE GROUP M4_GROUP
        OUTM4
        42-S0 42-S1
        37-D47 37-D67
        39-S0;
```

5

```
$
$ Simulation file for design 'TDM.sim', type '2064N48-50'
$ Created by PIN2LCA Ver. 1.01x at 10:58:00 JUL 14, 1987
$
!INPUT TDM.sim

GLOBALRESET-      .CLK 0 S0 1 S1 $ Initial pulse to reset latches
CK.PAD            .CLK 0 0 500 1 1000 0 .REP 0
LD.PAD            .CLK 0 0 40000 1
ADDR0.PAD         .CLK 0 0 40000 0 41000 1 42000 0 .REP 40000
+ 78000 0
ADDR1.PAD         .CLK 0 0 40000 0 42000 1 44000 0 .REP 40000
+ 78000 1
ADDR2.PAD         .CLK 0 0 40000 0 44000 1 48000 0 .REP 40000
+ 78000 1
ADDR3.PAD         .CLK 0 0 40000 0 48000 1 56000 0 .REP 40000
+ 78000 0
ADDR4.PAD         .CLK 0 0 40000 0 56000 1 72000 0 .REP 40000
+ 78000 1
D0.PAD            .CLK 0 1  1000 0          40000 0 41000 1
D1.PAD            .CLK 0 0  1000 1  2000 0 40000 1 41000 0 42000 1
D2.PAD            .CLK 0 0  2000 1  3000 0 40000 1 42000 0 43000 1
D3.PAD            .CLK 0 0  3000 1  4000 0 40000 1 43000 0 44000 1
D4.PAD            .CLK 0 0  4000 1  5000 0 40000 1 44000 0 45000 1
D5.PAD            .CLK 0 0  5000 1  6000 0 40000 1 45000 0 46000 1
D6.PAD            .CLK 0 0  6000 1  7000 0 40000 1 46000 0 47000 1
D7.PAD            .CLK 0 0  7000 1  8000 0 40000 1 47000 0 48000 1
D8.PAD            .CLK 0 0  8000 1  9000 0 40000 1 48000 0 49000 1
D9.PAD            .CLK 0 0  9000 1 10000 0 40000 1 49000 0 50000 1
D10.PAD           .CLK 0 0 10000 1 11000 0 40000 1 50000 0 51000 1
D11.PAD           .CLK 0 0 11000 1 12000 0 40000 1 51000 0 52000 1
D12.PAD           .CLK 0 0 12000 1 13000 0 40000 1 52000 0 53000 1
D13.PAD           .CLK 0 0 13000 1 14000 0 40000 1 53000 0 54000 1
D14.PAD           .CLK 0 0 14000 1 15000 0 40000 1 54000 0 55000 1
D15.PAD           .CLK 0 0 15000 1 16000 0 40000 1 55000 0 56000 1
D16.PAD           .CLK 0 0 16000 1 17000 0 40000 1 56000 0 57000 1
D17.PAD           .CLK 0 0 17000 1 18000 0 40000 1 57000 0 58000 1
D18.PAD           .CLK 0 0 18000 1 19000 0 40000 1 58000 0 59000 1
D19.PAD           .CLK 0 0 19000 1 20000 0 40000 1 59000 0 60000 1
D20.PAD           .CLK 0 0 20000 1 21000 0 40000 1 60000 0 61000 1
D21.PAD           .CLK 0 0 21000 1 22000 0 40000 1 61000 0 62000 1
D22.PAD           .CLK 0 0 22000 1 23000 0 40000 1 62000 0 63000 1 73000 0
D23.PAD           .CLK 0 0 23000 1 24000 0 40000 1 63000 0 64000 1
D24.PAD           .CLK 0 0 24000 1 25000 0 40000 1 64000 0 65000 1
D25.PAD           .CLK 0 0 25000 1 26000 0 40000 1 65000 0 66000 1
D26.PAD           .CLK 0 0 26000 1 27000 0 40000 1 66000 0 67000 1
D27.PAD           .CLK 0 0 27000 1 28000 0 40000 1 67000 0 68000 1
D28.PAD           .CLK 0 0 28000 1 29000 0 40000 1 68000 0 69000 1
D29.PAD           .CLK 0 0 29000 1 30000 0 40000 1 69000 0 70000 1
D30.PAD           .CLK 0 0 30000 1 31000 0 40000 1 70000 0 71000 1
D31.PAD           .CLK 0 0 31000 1 32000 0 40000 1 71000 0 72000 1

 .MONITOR CK.PAD LD.PAD ; ADDR4.PAD ADDR3.PAD ADDR2.PAD ADDR1.PAD ADDR0.PAD ; ;
+ C4 C3 C2 C1 C0 ;
+ D0.PAD D1.PAD D2.PAD D3.PAD D4.PAD D5.PAD D6.PAD D7.PAD ;
+ D8.PAD D9.PAD D10.PAD D11.PAD D12.PAD D13.PAD D14.PAD D15.PAD ;
+ D16.PAD D17.PAD D18.PAD D19.PAD D20.PAD D21.PAD D22.PAD D23.PAD ;
+ D24.PAD D25.PAD D26.PAD D27.PAD D28.PAD D29.PAD D30.PAD D31.PAD ; OUT.PAD

 .TABLE CK.PAD LD.PAD ; ADDR4.PAD ADDR3.PAD ADDR2.PAD ADDR1.PAD ADDR0.PAD ; ;
+ C4 C3 C2 C1 C0 ;
+ D0.PAD D1.PAD D2.PAD D3.PAD D4.PAD D5.PAD D6.PAD D7.PAD ;
+ D8.PAD D9.PAD D10.PAD D11.PAD D12.PAD D13.PAD D14.PAD D15.PAD ;
+ D16.PAD D17.PAD D18.PAD D19.PAD D20.PAD D21.PAD D22.PAD D23.PAD ;
+ D24.PAD D25.PAD D26.PAD D27.PAD D28.PAD D29.PAD D30.PAD D31.PAD ; OUT.PAD
```

```
*  P - S I L O S   1U.3  *   OUTPUTS      10:26:22     07-23-87

          CL AAAAA  CCCCC DDDDDDDD DDDDDDDD DDDDDDDD DDDDDDDD O
          KD DDDDD  43210 01234567 89111111 11112222 22222233 U
          .. DDDDD        ........ ..012345 67890123 45678901 T
          PP RRRRR        PPPPPPPP PP...... ........ ........ .
          AA 43210        AAAAAAAA AAPPPPPP PPPPPPPP PPPPPPPP P
          DD .....        DDDDDDDD DDAAAAAA AAAAAAAA AAAAAAAA A
             PPPPP                   DDDDDD DDDDDDDD DDDDDDDD D
             AAAAA
             DDDDD
 TIME
    0  00 00000  00000 10000000 00000000 00000000 00000000 1
  500  10 00000  00000 10000000 00000000 00000000 00000000 1
 1000  00 00000  00000 01000000 00000000 00000000 00000000 1
 1500  10 00000  00001 01000000 00000000 00000000 00000000 1
 2000  00 00000  00001 00100000 00000000 00000000 00000000 1
 2500  10 00000  00010 00100000 00000000 00000000 00000000 1
 3000  00 00000  00010 00010000 00000000 00000000 00000000 1
 3500  10 00000  00011 00010000 00000000 00000000 00000000 1
 4000  00 00000  00011 00001000 00000000 00000000 00000000 1
 4500  10 00000  00100 00001000 00000000 00000000 00000000 1
 5000  00 00000  00100 00000100 00000000 00000000 00000000 1
 5500  10 00000  00101 00000100 00000000 00000000 00000000 1
 6000  00 00000  00101 00000010 00000000 00000000 00000000 1
 6500  10 00000  00110 00000010 00000000 00000000 00000000 1
 7000  00 00000  00110 00000001 00000000 00000000 00000000 1
 7500  10 00000  00111 00000001 00000000 00000000 00000000 1
 8000  00 00000  00111 00000000 10000000 00000000 00000000 1
 8500  10 00000  01000 00000000 10000000 00000000 00000000 1
 9000  00 00000  01000 00000000 01000000 00000000 00000000 1
 9500  10 00000  01001 00000000 01000000 00000000 00000000 1
10000  00 00000  01001 00000000 00100000 00000000 00000000 1
10500  10 00000  01010 00000000 00100000 00000000 00000000 1
11000  00 00000  01010 00000000 00010000 00000000 00000000 1
11500  10 00000  01011 00000000 00010000 00000000 00000000 1
12000  00 00000  01011 00000000 00001000 00000000 00000000 1
12500  10 00000  01100 00000000 00001000 00000000 00000000 1
13000  00 00000  01100 00000000 00000100 00000000 00000000 1
13500  10 00000  01101 00000000 00000100 00000000 00000000 1
14000  00 00000  01101 00000000 00000010 00000000 00000000 1
14500  10 00000  01110 00000000 00000010 00000000 00000000 1
15000  00 00000  01110 00000000 00000001 00000000 00000000 1
15500  10 00000  01111 00000000 00000001 00000000 00000000 1
16000  00 00000  01111 00000000 00000000 10000000 00000000 1
16500  10 00000  10000 00000000 00000000 10000000 00000000 1
17000  00 00000  10000 00000000 00000000 01000000 00000000 1
17500  10 00000  10001 00000000 00000000 01000000 00000000 1
18000  00 00000  10001 00000000 00000000 00100000 00000000 1
18500  10 00000  10010 00000000 00000000 00100000 00000000 1
19000  00 00000  10010 00000000 00000000 00010000 00000000 1
19500  10 00000  10011 00000000 00000000 00010000 00000000 1
20000  00 00000  10011 00000000 00000000 00001000 00000000 1
20500  00 00000  10011 00000000 00000000 00001000 00000000 1
21000  00 00000  10100 00000000 00000000 00000100 00000000 1
21500  10 00000  10101 00000000 00000000 00000100 00000000 1
22000  00 00000  10101 00000000 00000000 00000010 00000000 1
22500  10 00000  10110 00000000 00000000 00000010 00000000 1
23000  00 00000  10110 00000000 00000000 00000001 00000000 1
23500  10 00000  10111 00000000 00000000 00000001 00000000 1
24000  00 00000  10111 00000000 00000000 00000000 10000000 1
24500  10 00000  11000 00000000 00000000 00000000 10000000 1
25000  00 00000  11000 00000000 00000000 00000000 01000000 1
25500  10 00000  11001 00000000 00000000 00000000 01000000 1
26000  00 00000  11001 00000000 00000000 00000000 00100000 1
26500  10 00000  11010 00000000 00000000 00000000 00100000 1
27000  00 00000  11010 00000000 00000000 00000000 00010000 1
27500  10 00000  11011 00000000 00000000 00000000 00010000 1
28000  00 00000  11011 00000000 00000000 00000000 00001000 1
28500  10 00000  11100 00000000 00000000 00000000 00001000 1
29000  00 00000  11100 00000000 00000000 00000000 00000100 1
29500  10 00000  11101 00000000 00000000 00000000 00000100 1
30000  00 00000  11101 00000000 00000000 00000000 00000010 1
30500  10 00000  11110 00000000 00000000 00000000 00000010 1
31000  00 00000  11110 00000000 00000000 00000000 00000001 1
31500  10 00000  11111 00000000 00000000 00000000 00000001 1
32000  00 00000  11111 00000000 00000000 00000000 00000000 1
32500  10 00000  00000 00000000 00000000 00000000 00000000 0
33000  00 00000  00000 00000000 00000000 00000000 00000000 0
39000  00 00000  00110 00000000 00000000 00000000 00000000 0
39500  10 00000  00111 00000000 00000000 00000000 00000000 0
40000  01 00000  00111 01111111 11111111 11111111 11111111 0
40500  11 00000  01000 01111111 11111111 11111111 11111111 0
41000  01 00001  01000 10111111 11111111 11111111 11111111 0
41500  11 00001  00001 10111111 11111111 11111111 11111111 0
42000  01 00010  00001 11011111 11111111 11111111 11111111 0
42500  11 00010  00010 11011111 11111111 11111111 11111111 0
```

5

```
          RD DDDDD  43210 01234567 89111111 11112222 22222233 U
          .. DDDDD  ..... ........ ..012345 67890123 45678901 T
          PP RRRRR        PPPPPPPP PP...... ........ ........ .
          AA 43210        AAAAAAAA AAPPPPPP PPPPPPPP PPPPPPPP P
          DD .....        DDDDDDDD DDAAAAAA AAAAAAAA AAAAAAAA A
                PPPPP                 DDDDDD DDDDDDDD DDDDDDDD D
                AAAAA
                DDDDD
 TIME
43000     01 00011  00010 11101111 11111111 11111111 11111111 0
43500     11 00011  00011 11101111 11111111 11111111 11111111 0
44000     01 00100  00011 11110111 11111111 11111111 11111111 0
44500     11 00100  00000 11110111 11111111 11111111 11111111 0
45000     01 00101  00000 11111011 11111111 11111111 11111111 0
45500     11 00101  00101 11111011 11111111 11111111 11111111 0
46000     01 00110  00101 11111101 11111111 11111111 11111111 0
46500     11 00110  00110 11111101 11111111 11111111 11111111 0
47000     01 00111  00110 11111110 11111111 11111111 11111111 0
47500     11 00111  00111 11111110 11111111 11111111 11111111 0
48000     01 01000  00111 11111111 01111111 11111111 11111111 0
48500     11 01000  01100 11111111 01111111 11111111 11111111 0
49000     01 01001  01100 11111111 10111111 11111111 11111111 0
49500     11 01001  01001 11111111 10111111 11111111 11111111 0
50000     01 01010  01001 11111111 11011111 11111111 11111111 0
50500     11 01010  01010 11111111 11011111 11111111 11111111 0
51000     01 01011  01010 11111111 11101111 11111111 11111111 0
51500     11 01011  01011 11111111 11101111 11111111 11111111 0
52000     01 01100  01011 11111111 11110111 11111111 11111111 0
52500     11 01100  01000 11111111 11110111 11111111 11111111 0
53000     01 01101  01000 11111111 11111011 11111111 11111111 0
53500     11 01101  01101 11111111 11111011 11111111 11111111 0
54000     01 01110  01101 11111111 11111101 11111111 11111111 0
54500     11 01110  01110 11111111 11111101 11111111 11111111 0
55000     01 01111  01110 11111111 11111110 11111111 11111111 0
55500     11 01111  01111 11111111 11111110 11111111 11111111 0
56000     01 10000  01111 11111111 11111111 01111111 11111111 0
56500     11 10000  10100 11111111 11111111 01111111 11111111 0
57000     01 10001  10100 11111111 11111111 10111111 11111111 0
57500     11 10001  10001 11111111 11111111 10111111 11111111 0
58000     01 10010  10001 11111111 11111111 11011111 11111111 0
58500     11 10010  10010 11111111 11111111 11011111 11111111 0
59000     01 10011  10010 11111111 11111111 11101111 11111111 0
59500     11 10011  10011 11111111 11111111 11101111 11111111 0
60000     01 10100  10011 11111111 11111111 11110111 11111111 0
60500     11 10100  10000 11111111 11111111 11110111 11111111 0
61000     01 10101  10000 11111111 11111111 11111011 11111111 0
61500     11 10101  10101 11111111 11111111 11111011 11111111 0
62000     01 10110  10101 11111111 11111111 11111101 11111111 0
62500     11 10110  10110 11111111 11111111 11111101 11111111 0
63000     01 10111  10110 11111111 11111111 11111110 11111111 0
63500     11 10111  10111 11111111 11111111 11111110 11111111 0
64000     01 11000  10111 11111111 11111111 11111111 01111111 0
64500     11 11000  11100 11111111 11111111 11111111 01111111 0
65000     01 11001  11100 11111111 11111111 11111111 10111111 0
65500     11 11001  11001 11111111 11111111 11111111 10111111 0
66000     01 11010  11001 11111111 11111111 11111111 11011111 0
66500     11 11010  11010 11111111 11111111 11111111 11011111 0
67000     01 11011  11010 11111111 11111111 11111111 11101111 0
67500     11 11011  11011 11111111 11111111 11111111 11101111 0
68000     01 11100  11011 11111111 11111111 11111111 11110111 0
68500     11 11100  11000 11111111 11111111 11111111 11110111 0
69000     01 11101  11000 11111111 11111111 11111111 11111011 0
69500     11 11101  11101 11111111 11111111 11111111 11111011 0
70000     01 11110  11101 11111111 11111111 11111111 11111101 0
70500     11 11110  11110 11111111 11111111 11111111 11111101 0
71000     01 11111  11110 11111111 11111111 11111111 11111110 0
71500     11 11111  11111 11111111 11111111 11111111 11111110 0
72000     01 00000  11111 11111111 11111111 11111111 11111111 0
72500     11 00000  00100 11111111 11111111 11111111 11111111 1
73000     01 00001  00100 11111111 11111111 11111101 11111111 1
73500     11 00001  00001 11111111 11111111 11111101 11111111 1
74000     01 00010  00001 11111111 11111111 11111101 11111111 1
74500     11 00010  00010 11111111 11111111 11111101 11111111 1
75000     01 00011  00010 11111111 11111111 11111101 11111111 1
75500     11 00011  00011 11111111 11111111 11111101 11111111 1
76000     01 00100  00011 11111111 11111111 11111101 11111111 1
76500     11 00100  00000 11111111 11111111 11111101 11111111 1
77000     01 00101  00000 11111111 11111111 11111101 11111111 1
77500     11 00101  00101 11111111 11111111 11111101 11111111 1
78000     01 10110  00101 11111111 11111111 11111101 11111111 1
78500     11 10110  10110 11111111 11111111 11111101 11111111 0
79000     01 10110  10110 11111111 11111111 11111101 11111111 0
79500     11 10110  10110 11111111 11111111 11111101 11111111 0
80000     01 10110  10110 11111111 11111111 11111101 11111111 0
```

Dual 32-bit Serial CRC Error Detection in an LCA Device

Karen Spesard

5

Abstract

The transmission and reception of digital data over local area networks (LANs) is more popular than ever. To transmit reliable information from one system to another over a LAN, an efficient error-detection technique is necessary. The error-detection scheme chosen by the IEEE for the Ethernet, a local area network developed by the Xerox Corporation, uses a 32-bit cyclic redundancy check (CRC).

A Dual 32-bit Serial CRC transmitter-receiver for the Ethernet is implemented in one Logic Cell™ Array (LCA device). The LCA device is a RAM-based CMOS circuit which is electrically programmable. It allows the designer to make design changes as necessary to accommodate other data communication protocols as well as other applications. This applications note describes the Ethernet CRC and how it is implemented in the versatile LCA device.

2175 Mission College Blvd. Santa Clara, CA 95054-1592 Tel: (408) 970-9700 TWX: 910-338-2376 TWX: 910-338-2374
Monolithic Memories

Dual 32-bit Serial CRC Error Detection in an LCA Device

Karen Spesard

Introduction

In transferring digital information from computer to computer, or from computer to peripheral devices, it is possible that errors in transmission may be introduced. The Cyclic Redundancy Check, or CRC, developed for the data communications industry will detect most of these errors. With one LCA device, a dual 32-bit serial CRC transmitter-receiver was implemented for the Ethernet, one of the industry's most popular local networks. This article describes the methodology used for designing the Dual 32-bit Serial CRC with an LCA device.

Overview of the CRC

Central to a serial hardware CRC system is a set of n-bit linear feedback shift registers, or LFSRs. These registers are designed to represent a fixed generator polynomial G(x). The generator polynomial is the divisor in equation 1,

$$x^n \frac{D(x)}{G(x)} = Q(x) + \frac{R(x)}{G(x)}, \quad (1)$$

and D(x) is the data polynomial, which is transmitted and checked for errors. The data polynomial is prescaled by x^n, the length of the generator polynomial, to insure the remainder is always different from the data itself.

The data is shifted in the LFSR and subsequently divided by G(x). The division generates the remainder polynomial R(x) but ignores the quotient polynomial Q(x). The remainder polynomial or redundancy check-bits remain in the registers after all of the data has been shifted in. The check-bits are then shifted out and appended to the data-bits to produce the encoded data. The encoded data becomes D(x) + R(x).

The encoded data is received in a similar LFSR with the same G(x). Because addition and subtraction operations are identical in modulo-2 arithmetic, equation (1) becomes:

$$Q(x)G(x) = x^nD(x) + R(x). \quad (2)$$

Since D(x) was prescaled, and appending the remainder to the data is equivalent to adding it, the new encoded polynomial should be exactly divisible by G(x). Thus, if the remainder of this operation is zero, all of the original transmitted data-bits are assumed unaltered. If the remainder is anything other than zero, an error occurred and an error flag is set.

Generating CRC Bits and Checking Data

An example of the LFSR hardware needed to generate CRC bits serially for a three-bit G(x) is shown in Figure 1. In this example, the output of the last register is XORed, or shift subtracted, with the data-bit before feeding back to registers X0 and X2. The feedback term positions in the LFSR correspond to all but the highest power of x in the generator polynomial. In this case, the feedback positions are determined from x^2 and x^0.

The redundancy check-bits can be calculated from Table 1 for $G(x) = x^3 + x^2 + 1$ where the internal state of each register in the LFSR after every shift is shown. Therefore, if the data is 10011, the redundancy check-bits are D5 = 0, D6 = 1, and D7 = 0 after the last data-bit is shifted in (5th shift). This is because D0 = 1, D1 = 1, D2 = 0, D3 = 0, and D4 = 1. ($D(x) = 1 + 0x + 0x^2 + 1x^3 + 1x^4$ and the least significant bit, LSB, is the first bit transmitted.) The check-bits are shifted out from the LFSR by

REGISTERS	X0	X1	X2	COMMENTS
Initialize	0	0	0	If registers initialized to 0
1st shift	D0	0	D0	
2nd shift	D0⊕D1	D0	D0⊕D1	
3rd shift	D0⊕D1⊕D2	D0⊕D1	D1⊕D2	
4th shift	D1⊕D2⊕D3	D0⊕D1⊕D2	D0⊕D2⊕D3	
5th shift	D0⊕D2⊕D3⊕D4	D1⊕D2⊕D3	D1⊕D3⊕D4	Redundancy check bits
6th shift	D1⊕D3⊕D4⊕D5	D0⊕D2⊕D3⊕D4	D2⊕D4⊕D5	
7th shift	D2⊕D4⊕D5⊕D6	D1⊕D3⊕D4⊕D5	D0⊕D3⊕D5⊕D6	
8th shift	D0⊕D3⊕D5⊕D6⊕D7	D2⊕D4⊕D5⊕D6	D0⊕D1⊕D4⊕D6⊕D7	Zero remainder

⊕ = EXOR

Table 1. Calculation of Redundancy Check-Bits for $G(X) = X^3 + X^2 + 1$

disabling the feedback terms of the LFSR. The check-bits can be verified by long division as in Figure 2.

The encoded data then becomes 010 10011. To check for data integrity, the encoded data is shifted in another LFSR with the same G(x). This time the last bit (8th bit) is shifted in, and if the message was unaltered, the bits residing in the registers are zero. If the redundancy bits are not zero, an error occurred. This can also be verified by long division as in Figure 3.

In summary, CRC modifies the data polynomial so that it is exactly divisible by a fixed polynomial G(x). When the modified polynomial is received, it is checked for the exact division by G(x). If an error occurred, the RDYFLG is set.

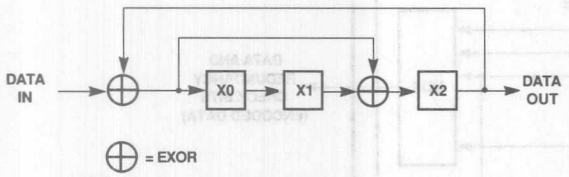


Figure 1. LFSR for G(X) = $X^3 + X^2 + 1$

```
        1001
     ----------
1101 | 11001000   PRESCALED DATA
       1101       WITH 3 ZERO BITS
     ----------
        11000     REMAINDER
        11010
     ----------
          010
```

Figure 2. Long Division Example - With Pure Data

```
        1001
     ----------
1101 | 11001010   ENCODED DATA
       1101
     ----------
        011010
        11010
     ----------
            0     NO REMAINDER
```

Figure 3. Long Division Example - With Encoded Data

Why Use LCA Devices in CRC?

The 32-bit serial CRC, with transmission and reception sections, can be implemented in several ways. One way would be to acquire standard CRC chips. These chips, however, only exist with specific G(x) and therefore can be inflexible. Another method would be to design the transmission and reception sections of the CRC with traditional PLD devices. This would require eight PLD devices for the dual 32-bit LFSRs (eight bits in each PLD) even without additional control logic. A third method would use an LCA device, as it would take only one to implement the dual 32-bit CRC and with more flexibility than a dedicated custom part.

In addition, the LCA device can provide designers with more control after design release than the alternative solutions. For example, the designer can change the generator polynomial or the control logic "on-the-fly" by merely reprogramming the configuration data from separate sections of an EPROM. A large EPROM, 8Kx8 or 16Kx8, can typically store three and six configuration patterns, respectively.

32-Bit CRC Design

CRC error detection is one of the best methods for checking the validity of large frames of information. It can detect all errors within n successive bits, all errors with an odd number of bits in error for even G(x), and, of course, all error patterns that are not divisible by G(x).

The IEEE-802.3/Ethernet Local Area Network Standard defines a 32-bit CRC code. This code works with data ranges from 46 to 1500 bytes, and is also used in the Autodin-II Network.

The generator polynomial used for the Ethernet 32-bit CRC is defined as:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1,$$

where the coefficients of the generator polynomial correspond to the feedback positions in the LFSR.

Figure 4 shows the block diagram of the Dual 32-bit Serial CRC. As the diagram illustrates, the transmitter portion of the CRC is controlled separately from the receiver portion.

The standard specifies that on the transmitter end, the shift registers are preloaded to ones with INITA, and CONTROLA is held HIGH while the incoming data bits are shifted in to generate the CRC. When all of the data bits have been entered, CONTROLA is held LOW and the complemented CRC is shifted out for transmission.

On the receiver end, the data bits and the complemented check bits are sent. When the end of the frame is reached, the remainder is checked. If no error occurs, the final contents of the shift register are:

X31 X0

11000111 00000100 11011101 01111011

This remainder is not 0 because the check-bits are complemented before being transmitted to the receiver.

CRC Logic Implementation

Figures 5 and 6 show the complete design for the 32-bit CRC with the standard generator polynomial described above. The generator polynomial is implemented with thirty-two registers in each LFSR. The feedback terms (Q31 xor DIN) are at registers Q0, Q1, Q2, Q4, Q5, Q7, ... Q23, and Q26 and correspond to the Ethernet generator polynomial, G(x) given above.

The registers in the LFSRs are D-type flip-flops and are clocked synchronously. In the transmitter portion, two pins - CONTROL and INIT - are respectively added for initializing the registers and controlling the feedback terms. When INIT is HIGH, all registers are initialized to HIGH. When CONTROL is HIGH, the multiplexer shifts out data bits and generates check-bits. When CONTROL is LOW, the feedback term is disabled and inverted redundancy check-bits are shifted out from the LFSR.

The five-bit synchronous ripple carry counter is enabled with the control line LOW and reset with the control line HIGH. When the five count bits reach all ones, the thirty-two check bits are shifted out of the LFSR, and the RDYFLAG is set.

The receiver LFSR is essentially the same as the transmitter LFSR. However, after data and the thirty-two check bits are received, the data is checked for errors. If an error occurred, the error flag is set.

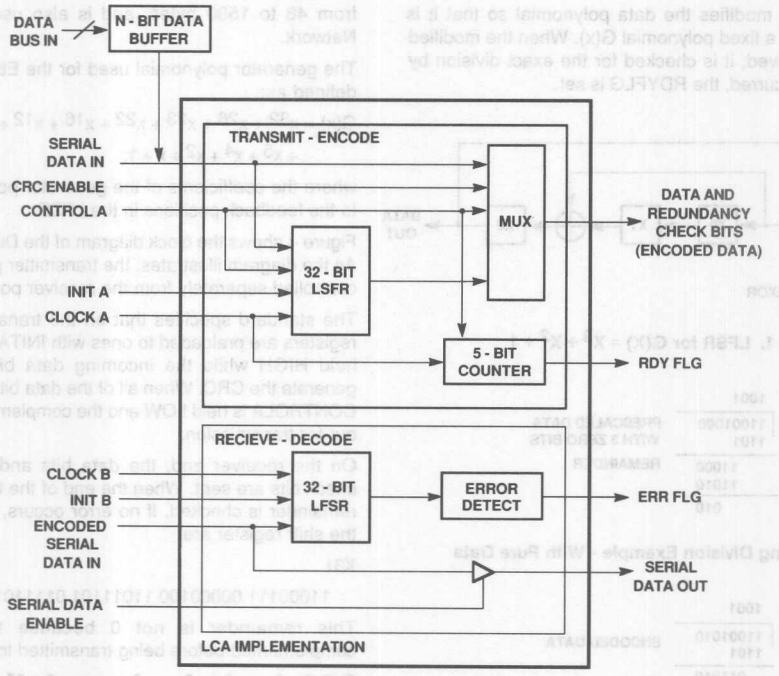


Figure 4. Block Diagram of the Dual 32-Bit Serial CRC

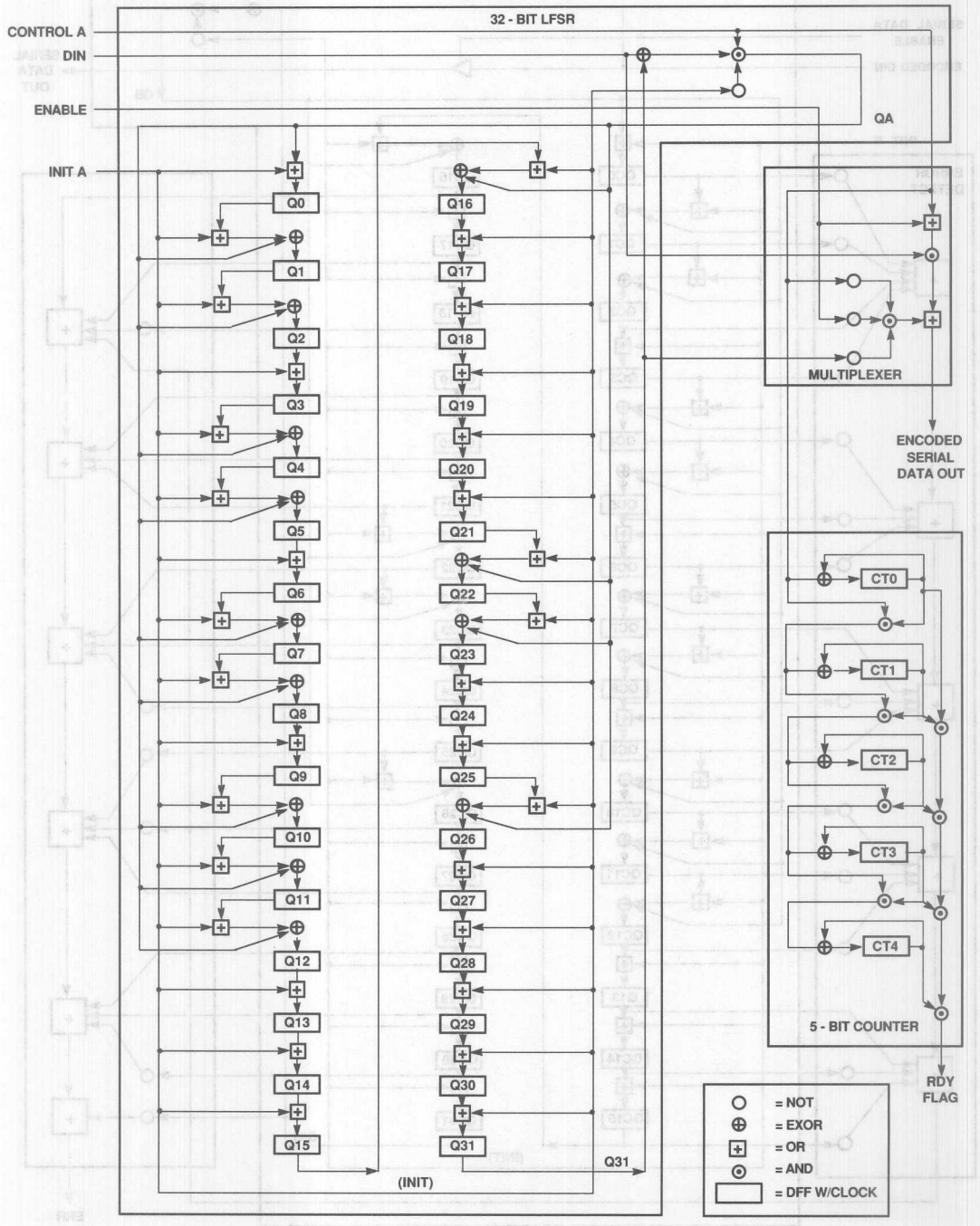


Figure 5. Logic Implementation of Transmitter-Encoder Section

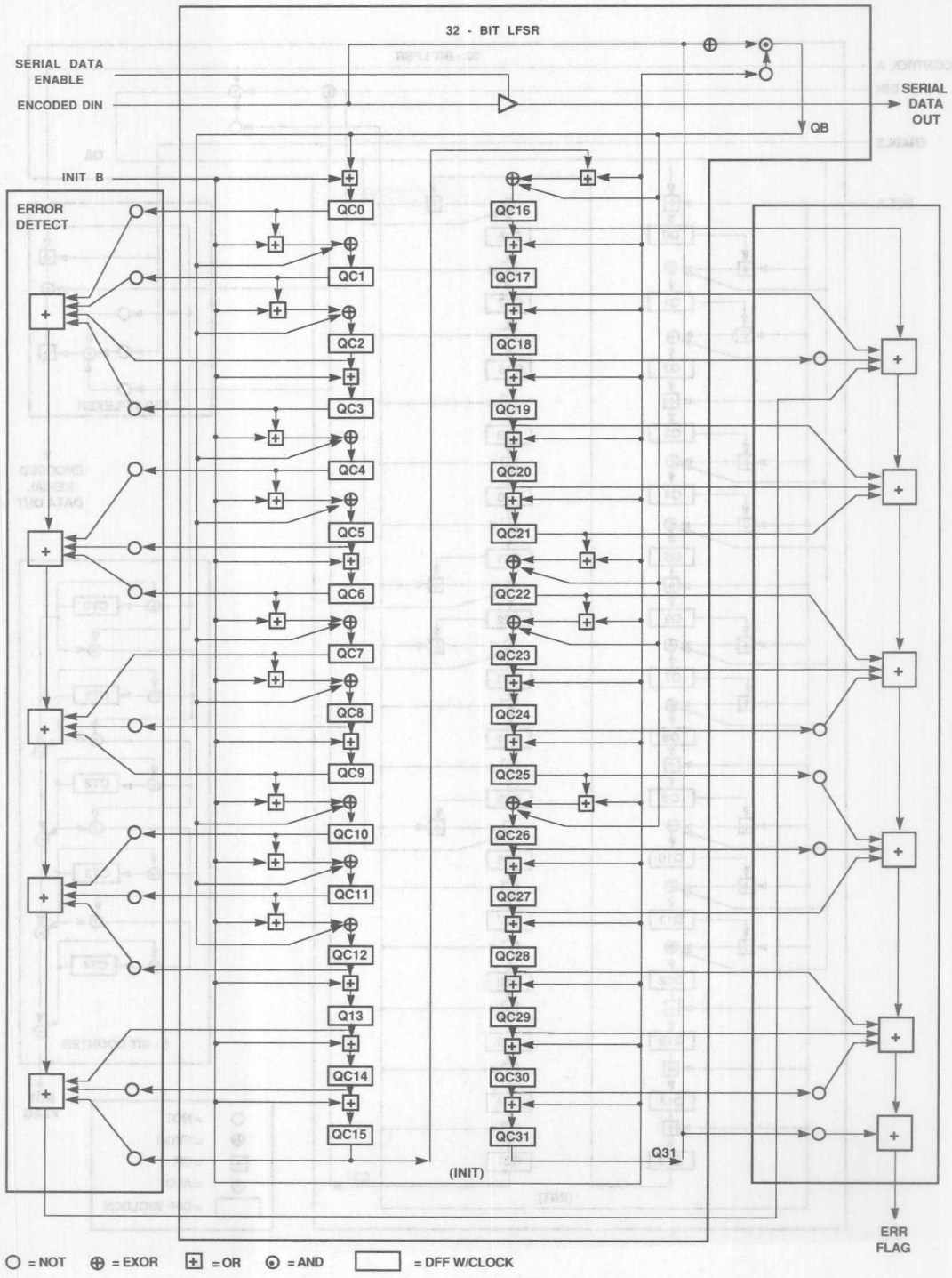


Figure 6. Logic Implementation of Receiver-Decoder Section

Logic Cell Array Implementation

The Dual 32-bit Serial CRC circuit described above is implemented in an M2018 LCA device. It contains one hundred configurable logic blocks (CLBs) and seventy-four user-configurable input/output blocks (IOBs) providing a total of 1800 usable logic gates. The logic is partitioned and placed in a graphics environment on the IBM® PC-XT™ or PC-AT™ using the Monolithic Memories' XACT™ LCA Editor.

The Monolithic Memories' XACT LCA Editor provides the tools from which the logic and interconnects (CLBs, IOBs, and programmable interconnect points) are constructed. The configuration of specified logic blocks can be made with the XACT "Edit Block" command. Once in the "Edit Block" menu, a combinational, latched, or registered equation can be selected with the mouse. The clock input polarity, set and reset functions can also be selected as required. The CLB can be configured as one output function of four Boolean input variables, or two functions of three Boolean input variables. The logic equations can be input from the PC keyboard using Boolean algebra, or from the mouse using a Karnaugh map. For an IOB, the "Edit Block" menu can be used to select pad and/or latched inputs, buffered and/or three-state control outputs, or bidirectional I/Os.

The programmable interconnect points (PIPs), are configured using the XACT "NET" and "PIN" menus. The features used from the "NET" and "PIN" menus are summarized below.

Command	Function
Name Net	Names a Net
Add Net	Names and Adds a Net
Add Pin	Adds Net and Automatically Routes a Net
Edit Net	Manually Routes a Net
Route Net	Routes Named Nets

When the nets are routed, the PIPs are programmed automatically.

An example of a CLB is shown in Figure 7. Here the output of the CLB is combinational, and the CLB is configured as a multiplexer. The multiplexer selects the serial data when CONTROL or ENABLE are HIGH and selects $\overline{Q31}$ when CONTROL and ENABLE are LOW. The Karnaugh map and the Boolean equation in the figure illustrate the logic used to implement this function. The output of the multiplexer is named ENCDATAOUT.

Figure 8 shows the configuration of the CLBs for the modulo-2 addition of the feedback signal with the data, when INIT is LOW and CONTROL is HIGH. The output again is combinational and is named QA.

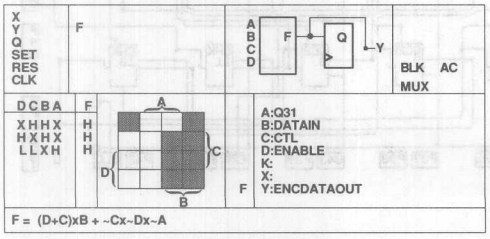


Figure 7. Mux for Data/Check Bit Select

Figures 9 to 11 show the configuration used to implement the logic for the first few shift registers, corresponding to equations in Tables 2 and 3. Note that these shift registers are configured as registered equations. Figure 12 illustrates the signal configuration of the RDYFLG in the transmitter section

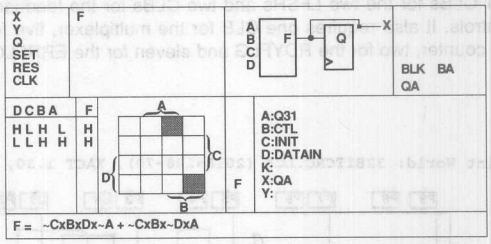


Figure 8. Modulo-2 Addition of Q31 and DATAIN

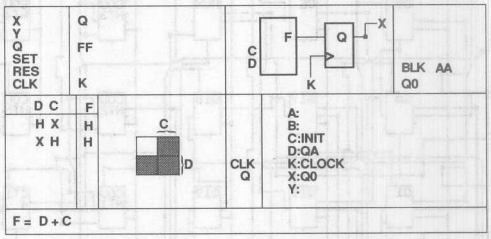


Figure 9. The 1st Bit of the LFSR

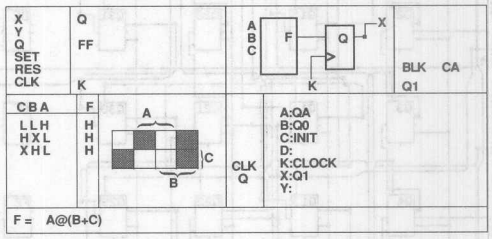


Figure 10. The 2nd Bit of the LFSR

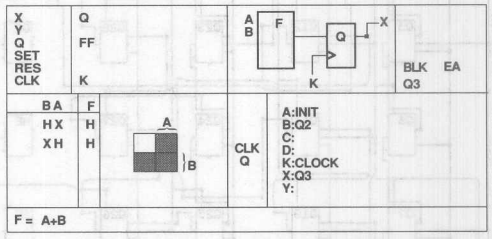


Figure 11. The 4th Bit of the LFSR

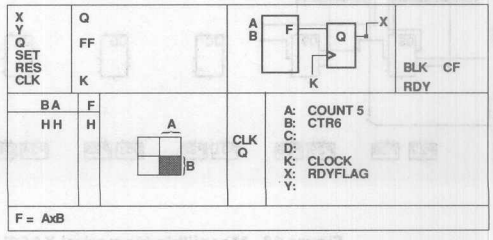


Figure 12. The RDY Flag Signal

5

CTR14 is an AND gate of input bits 1 - 4.

The placed and routed Dual 32-bit CRC design requires sixty-four CLBs for the two LFSRs and two CLBs for the feedback controls. It also requires one CLB for the multiplexer, five for the counter, two for the RDYFLG and eleven for the ERRFLG.

chosen for this implementation. This 2018NL68 has a speed grade of 70 MHz, the flip-flop toggle rate. A copy of the Monolithic Memories' XACT EDITLCA file is shown in Figure 13 with all of the CLBs and IOBs labelled.

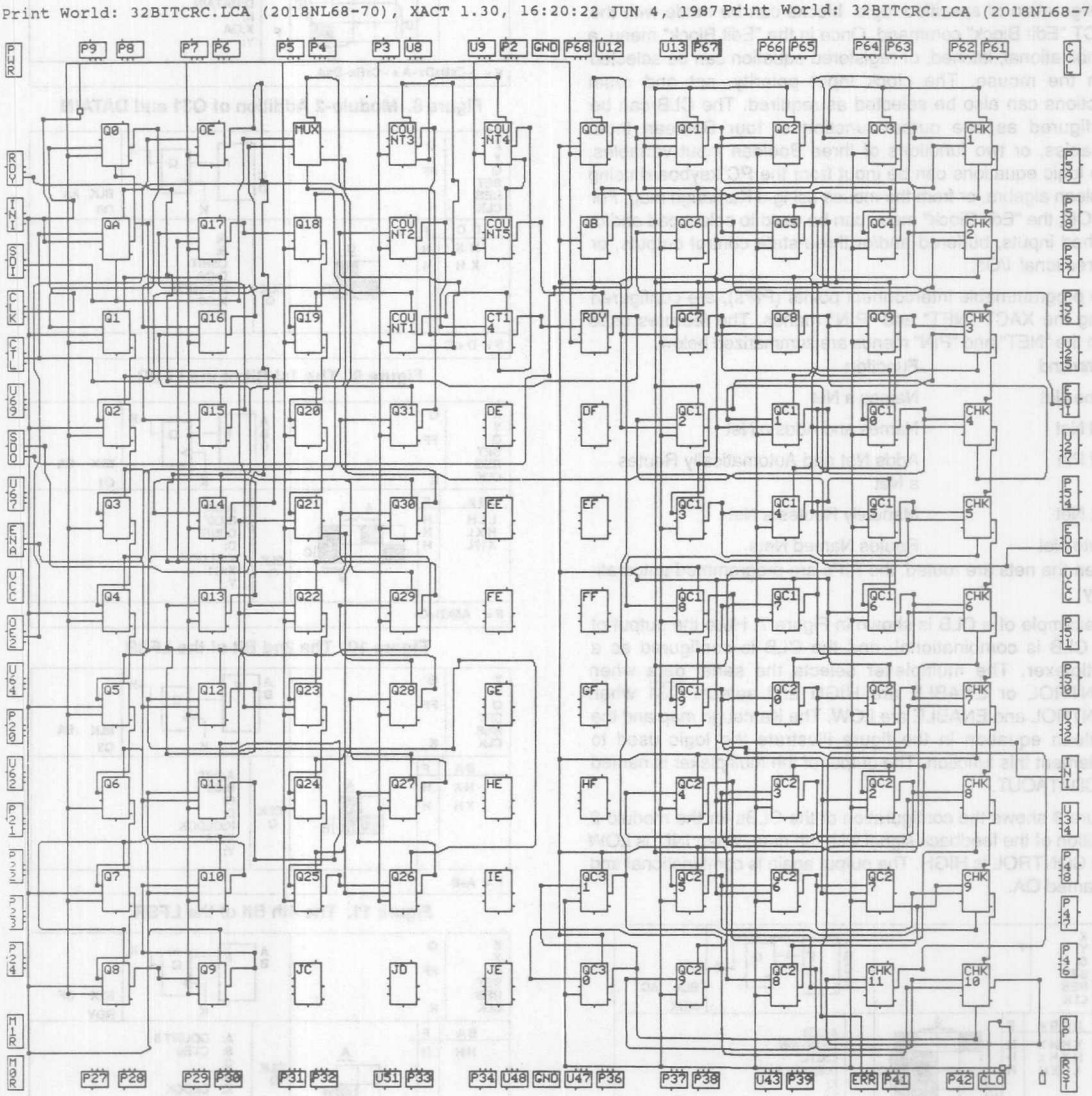


Figure 13. Monolithic Memories' XACT "EDITLCA" File for the Dual 32-Bit Serial CRC

EQUATIONS FOR 32-BIT SERIAL CRC
TRANSMISSION SECTION

```
QA  =  CTL*/INIT(Q31@DIN)      ;MOD 2 ADDITION, IF CTL*/INIT
Q0  := QA+INIT                 ;SHIFT QA OR INITIALIZE
Q1  := (Q0+INIT)@QA            ;SHIFT Q0@QA OR INITIALIZE
Q2  := (Q1+INIT)@QA            ;SHIFT Q1@QA OR INITIALIZE
Q3  := Q2+INIT                 ;SHIFT Q2 OR INITIALIZE
Q4  := (Q3+INIT)@QA            ;SHIFT Q3@QA OR INITIALIZE
Q5  := (Q4+INIT)@QA            ;SHIFT Q4@QA OR INITIALIZE
Q6  := Q5+INIT                 ;SHIFT Q5 OR INITIALIZE
Q7  := (Q6+INIT)@QA            ;SHIFT Q6@QA OR INITIALIZE
Q8  := (Q7+INIT)@QA            ;SHIFT Q7@QA OR INITIALIZE
Q9  := Q8+INIT                 ;SHIFT Q8 OR INITIALIZE
Q10 := (Q9+INIT)@QA            ;SHIFT Q9@QA OR INITIALIZE
Q11 := (Q10+INIT)@QA           ;SHIFT Q10@QA OR INITIALIZE
Q12 := (Q11+INIT)@QA           ;SHIFT Q11@QA OR INITIALIZE
Q13 := Q12+INIT                ;SHIFT Q12 OR INITIALIZE
Q14 := Q13+INIT                ;SHIFT Q13 OR INITIALIZE
Q15 := Q14+INIT                ;SHIFT Q14 OR INITIALIZE
Q16 := (Q15+INIT)@QA           ;SHIFT Q15@QA OR INITIALIZE
Q17 := Q16+INIT                ;SHIFT Q16 OR INITIALIZE
Q18 := Q17+INIT                ;SHIFT Q17 OR INITIALIZE
Q19 := Q18+INIT                ;SHIFT Q18 OR INITIALIZE
Q20 := Q19+INIT                ;SHIFT Q19 OR INITIALIZE
Q21 := Q20+INIT                ;SHIFT Q20 OR INITIALIZE
Q22 := (Q21+INIT)@QA           ;SHIFT Q21@QA OR INITIALIZE
Q23 := (Q22+INIT)@QA           ;SHIFT Q22@QA OR INITIALIZE
Q24 := Q23+INIT                ;SHIFT Q23 OR INITIALIZE
Q25 := Q24+INIT                ;SHIFT Q24 OR INITIALIZE
Q26 := (Q25+INIT)@QA           ;SHIFT Q25@QA OR INITIALIZE
Q27 := Q26+INIT                ;SHIFT Q26 OR INITIALIZE
Q28 := Q27+INIT                ;SHIFT Q27 OR INITIALIZE
Q29 := Q28+INIT                ;SHIFT Q28 OR INITIALIZE
Q30 := Q29+INIT                ;SHIFT Q29 OR INITIALIZE
Q31 := Q30+INIT                ;SHIFT Q30 OR INITIALIZE

ENCDATAOUT = (ENABLE+CTL)*SERDATAIN+(/ENABLE*/CTL*Q31)
RDYFLG := COUNT5*CTR14
```

Table 2.

EQUATIONS FOR 32-BIT SERIAL CRC
RECEPTION SECTION

```
QB    =  /INIT*(Q31@DIN)          ;MOD 2 ADDITION, IF /INIT
QC0   := QB+INIT                  ;SHIFT QB OR INITIALIZE
QC1   := (QC0+INIT)@QB            ;SHIFT QC0@QB OR INITIALIZE
QC2   := (QC1+INIT)@QB            ;SHIFT QC1@QB OR INITIALIZE
QC3   := QC2+INIT                 ;SHIFT QC2 OR INITIALIZE
QC4   := (QC3+INIT)@QB            ;SHIFT QC3@QB OR INITIALIZE
QC5   := (QC4+INIT)@QB            ;SHIFT QC4@QB OR INITIALIZE
QC6   := QC5+INIT                 ;SHIFT QC5 OR INITIALIZE
QC7   := (QC6+INIT)@QB            ;SHIFT QC6@QB OR INITIALIZE
QC8   := (QC7+INIT)@QB            ;SHIFT QC7@QB OR INITIALIZE
QC9   := QC8+INIT                 ;SHIFT QC8 OR INITIALIZE
QC10  := (QC9+INIT)@QB            ;SHIFT QC9@QB OR INITIALIZE
QC11  := (QC10+INIT)@QB           ;SHIFT QC10@QB OR INITIALIZE
QC12  := (QC11+INIT)@QB           ;SHIFT QC11@QB OR INITIALIZE
QC13  := QC12+INIT                ;SHIFT QC12 OR INITIALIZE
QC14  := QC13+INIT                ;SHIFT QC13 OR INITIALIZE
QC15  := QC14+INIT                ;SHIFT QC14 OR INITIALIZE
QC16  := (QC15+INIT)@QB           ;SHIFT QC15@QB OR INITIALIZE
QC17  := QC16+INIT                ;SHIFT QC16 OR INITIALIZE
QC18  := QC17+INIT                ;SHIFT QC17 OR INITIALIZE
QC19  := QC18+INIT                ;SHIFT QC18 OR INITIALIZE
QC20  := QC19+INIT                ;SHIFT QC19 OR INITIALIZE
QC21  := QC20+INIT                ;SHIFT QC20 OR INITIALIZE
QC22  := (QC21+INIT)@QB           ;SHIFT QC21@QB OR INITIALIZE
QC23  := (QC22+INIT)@QB           ;SHIFT QC22@QB OR INITIALIZE
QC24  := QC23+INIT                ;SHIFT QC23 OR INITIALIZE
QC25  := QC24+INIT                ;SHIFT QC24 OR INITIALIZE
QC26  := (QC25+INIT)@QB           ;SHIFT QC25@QB OR INITIALIZE
QC27  := QC26+INIT                ;SHIFT QC26 OR INITIALIZE
QC28  := QC27+INIT                ;SHIFT QC27 OR INITIALIZE
QC29  := QC28+INIT                ;SHIFT QC28 OR INITIALIZE
QC30  := QC29+INIT                ;SHIFT QC29 OR INITIALIZE
QC31  := QC30+INIT                ;SHIFT QC30 OR INITIALIZE

IF /OE2, SERDATAOUT := ENCDATAIN

ERRFLG := /Q0+/Q1+Q2+/Q3+/Q4+/Q5+/Q6+Q7+/Q8+Q9+/Q10+/Q11+/Q12+Q13
          +/Q14+/Q15+Q16+Q17+/Q18+Q19+Q20+Q21+A22+Q23+/Q24+/Q25
          +/Q26+Q27+Q28+Q29+/Q30+Q31
```

Table 3.

Configuration

The LCA device is manufactured with a programmable RAM-based CMOS process and must be configured upon power-up. This requires the loading of a configuration program into the LCA device. A number of methods for doing this exist. The Master Mode LOW was selected in this application.

In this mode, the LCA device is configured from an external EPROM containing the configuration data as shown in Figure 14. After an initial power-up delay and with the $\overline{\text{RESET}}$ pin HIGH, the LCA device reads the byte-wide configuration data from the EPROM starting at address 0000 and loads it into pins D0-D7. When the DONE pin goes HIGH, configuration is complete and the LCA device is free to perform as designed. The configuration process, lasts approximately 25 ms.

Summary

In this applications note, the principles of designing a Dual 32-bit Serial CRC in a Monolithic Memories' LCA device was described. The device implementation has many advantages which include its user-configurable logic functions, interconnects and I/O pins. This flexibility facilitates the design process significantly as standards in the data communications industry continually change. Thus, the LCA device is a viable solution for designers who wish to gain more control over their designs.

The Dual 32-bit CRC design developed for the LCA device is available upon request. The bit pattern and .LCA file will be provided for programming the LCA device in an EPROM. Please ask for design XDES09.LCA.

References

1. Nadia Sachs, *"Cyclic Redundancy Check using PAL® Devices."* AN-105, Monolithic Memories Inc., 2175 Mission College Blvd., Santa Clara, CA 95054-1592
2. Vivian Kong, *"Implementation of Serial/Parallel CRC Using PAL Devices."* AN-125, Monolithic Memories Inc., 2175 Mission College Blvd., Santa Clara, CA 95054-1592
3. IEEE Std 802.3-1985 *Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications,* The Institute of Electrical and Electronics Engineers, Inc., Wiley-Interscience, 1985
4. Robert Swanson, *"Understanding Cyclic Redundancy Codes,"* Computer Design, Nov. 1975.

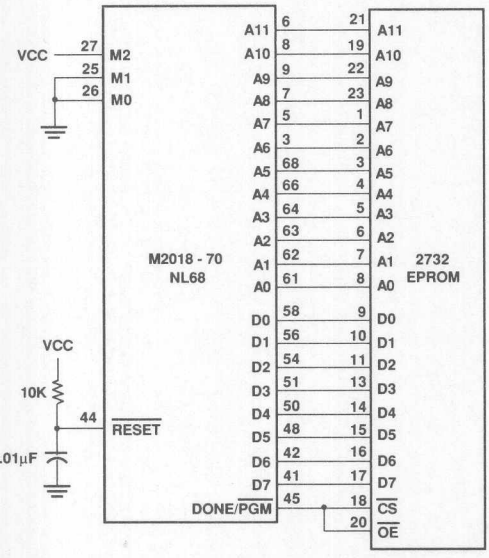


Figure 14. Master Mode Low Configuration for the Dual 32-Bit Serial CRC

5

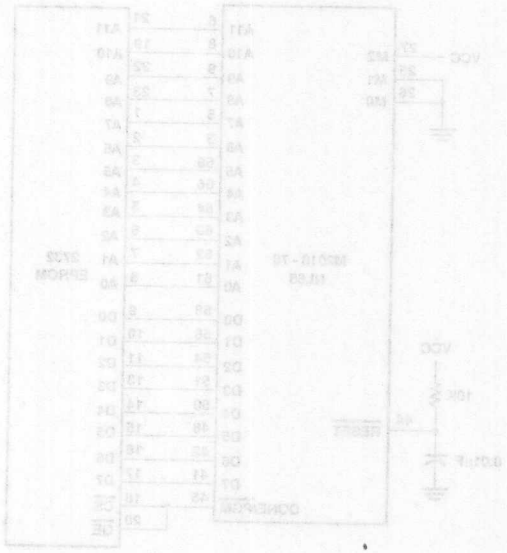


Figure 14. Master Mode Low Configuration for the Dual 32-Bit Serial CRC

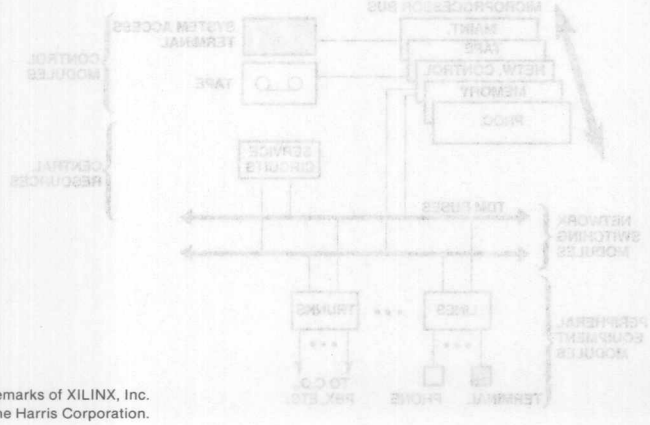
LCA Device Implements an 8-Bit Format Converter in a PBX Switching Module

Raj Paripatyadar and C.B. Lee

Abstract

In Private Branch Exchanges, thousands of voice channels are multiplexed using time division multiplexing (TDM) techniques. Many of these voice channels may be multiplexed onto either serial or parallel channels. This application note describes an eight-bit format converter (parallel-to-serial and serial-to-parallel) circuit which transposes eight serial TDM highways into a single eight-bit parallel bus. A user-programmable Monolithic Memories' Logic Cell™ Array (LCA device) implements this circuit very efficiently using ninety-nine Configurable Logic Blocks (CLBs) out of one hundred CLBs available. Each CLB has a combinational logic section and a storage element. The circuit operates at up to 18.5 MHz when implemented in a 70-MHz CMOS, LCA M2018 device

5



2175 Mission College Blvd. Santa Clara, CA 95054-1592 Tel: (408) 970-9700 TWX: 910-338-2376
TWX: 910-338-2374
Monolithic Memories MMI

LCA Device Implements an 8-Bit Format Converter in a PBX Switching Module

Raj Paripatyadar and C.B. Lee

Introduction

In a Private Branch Exchange (PBX) or Central Office (CO), incoming digitized voice circuits are "switched" or routed to their appropriate destination. The "switching" takes place while the data is in serial or parallel format. An eight-bit format converter circuit, widely used in a variety of PBX and CO architectures, translates the serial data streams into parallel data streams and vice-versa.

The eight-bit format converter circuit requires a large number of register elements. These elements, namely serial shift registers, can be implemented efficiently in an LCA device due to its high register content and flexible structure. The LCA M2018 programmable device contains one hundred configurable register elements, ninety-nine of which are used in the implementation of the converter circuit. This circuit is capable of running at 12.5 MHz and 18.5 MHz in a 50-MHz and 70-MHz LCA device, respectively.

Powerful software tools and circuit uniformity allowed the design to be laid out and simulated in just two days. However, if different support logic becomes necessary, PBX vendors are able to generate a new configuration program in the LCA reprogrammable device. Since these devices are manufactured in a CMOS technology, a complete PBX system with several LCA devices can help keep the active power consumption down. (Please refer to the LCA Design and Applications Handbook-Reference 3 for basic information regarding structure and programming aspects.)

Overview of a PBX

Central Offices (CO) and Private Branch Exchanges (PBX) provide both telephone and data services. A PBX may be considered as a localized telephone exchange serving the interconnection requirements of users in office environments. Outgoing calls are directed out of the PBX and routed through the local CO to the far destination. In general terms, a PBX or a CO consists of major blocks shown in Figure 1. Racks of peripheral equipment modules containing line cards provide access to telephone units, or to data adapter modules which provide data communication services. Trunk cards provide high-speed links to host computers, or other PBXs. Network switching modules interconnect calling parties to answering parties via digital multiplexed links. Control modules containing a Central Processor Unit (CPU), mass storage and local memory, perform the "setup" and "tear-down" of each circuit connection, as well as the monitoring of PBX activities. Central resources are modules of hardware and software which operate in a time-shared manner. The central resources include facilities such as ringing tone generator and message recording modules.

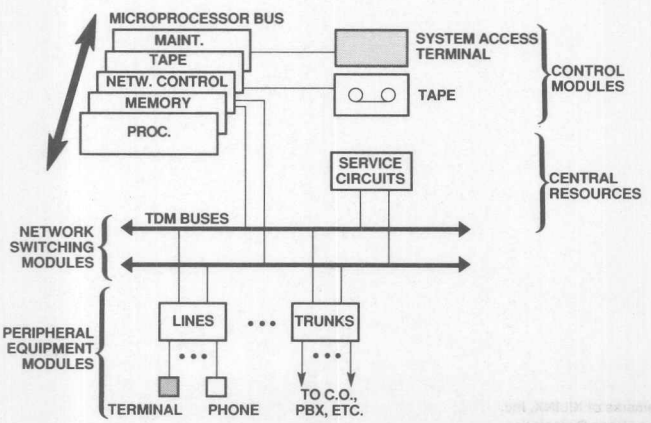


Figure 1. PBX Communication and Control Architecture

Hierarchy of Channel Multiplexing in a PBX

The main function of switching modules is to interconnect circuits between two or more parties such as in a conference call. A digital telephone unit, or a line card, contains a Coder/Decoder (CODEC) device. The CODEC device digitizes voice and transmits it into a Pulse Code Modulated (PCM) data stream on a common serial data highway at appropriate time intervals, which are assigned at the start of the call.

Analog voice is sampled at 8 KHz and passes through an eight-bit analog-to-digital converter. The digitized sample (eight-bit binary word) is transmitted serially to produce a serial data stream of (8 x 8 =) 64 Kbps. Thirty-two of these samples or CODECs are normally connected to a common highway operating at (32 x 64 =) 2.048 Mbps. Each CODEC is assigned "Serial Time Slots" of eight-bit duration, and sends its eight bits of data at the rate of 2.048 Mbps every 8 KHz. Another time slot is available on a separate highway for the receive data stream.

Therefore, a 2.048 Mbps highway can support thirty-two channels or voice connections in one direction. However, there are several ways to combine serial highways for additional voice connections. One way is to combine four highways into one high-speed highway of 8.192 Mbps. Further integration is possible, but the aggregate data rate becomes high and leads to implementation problems including timing, signal reflection, and radiation, to name a few.

Another way to integrate channels without increasing the clock rate is to convert eight serial data highways into a single eight-bit parallel bus. With this method, the serial clock rate and the parallel clock rate are the same, 2.048 Mbps. This means that data throughput on the eight serial highways of 16.384 Mbps is maintained in the eight-bit parallel bus operating at 2.048 Mbps. This works by careful alignment of the incoming parallel time slots to serial time slots on the serial highways. An eight-bit format converter scheme, sometimes called a "Corner Bender", is widely used in PBX or CO switching modules. Four of these parallel buses are combined (see Figure 2) to form a single parallel system bus that is thirty-two bits wide, yet still operates at 2.048 Mbps. The multiplexing architecture shown in Figure 2 is used in the Harris 20-20™ Integrated Network Switch (see Reference 1).

Space switching and/or time slot switching to interconnect different parties is performed either in the serial highway format or in the parallel data stream format.

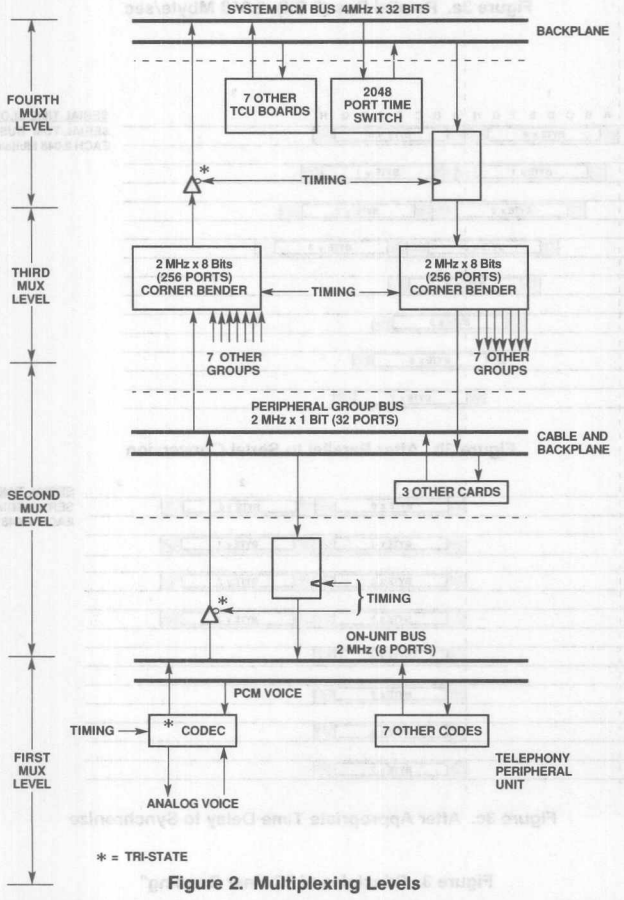


Figure 2. Multiplexing Levels

The circuit consists of eight parallel-to-serial shift registers, and additional shift registers which provide proportional delays to synchronize the outputs. The principle of operation is shown in Figure 3, and the circuit is shown in Figure 4. Parallel data from the parallel bus is loaded with each clock into one of the eight shift registers in a rotating pattern. The data then shifts out of the shift register with each subsequent clock. After all eight registers have been loaded, the first register will be empty on the next clock. New data is loaded into this register at the same clock edge as the last bit shifts out to the next stage. Thus, a continuous flow of data through the registers is insured. The data out of the eight shift registers is skewed in relation to each other as shown in Figure 3b. The various space slots on the highways need to be aligned to keep synchronization. This is achieved by proportionally delaying the outputs. Channel 1 output is delayed by seven clocks (using a seven-bit serial shift register), channel 2 is delayed by six clocks, etc. Channel 8 output does not go through any additional delay. The final outputs of the delay registers are aligned and data is clocked out in a synchronous fashion. See Figure 3c.

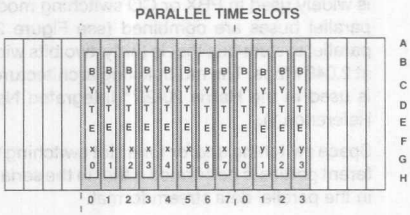


Figure 3a. Parallel Bus (8 Bit), 2.048 Mbyte/sec

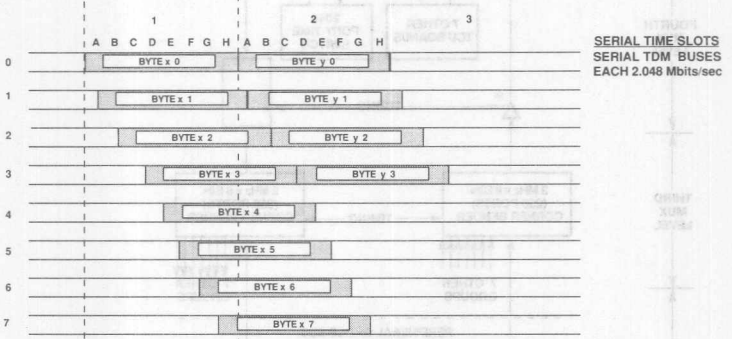


Figure 3b. After Parallel to Serial Conversion

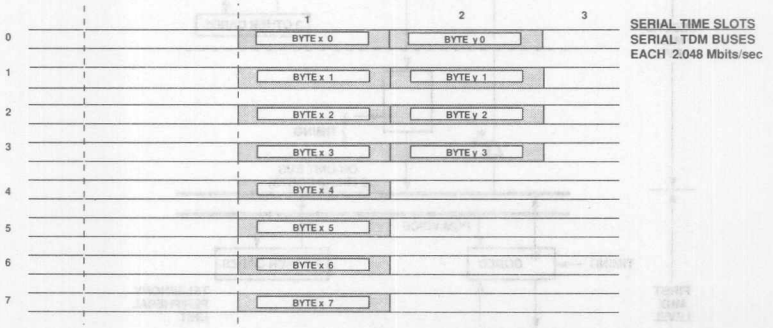


Figure 3c. After Appropriate Time Delay to Synchronize

Figure 3. Principles of "Corner Bending"

Figure 4 also shows the additional circuitry needed to control the shift registers. A preloadable three-bit counter keeps count of eight clock pulses. The parallel-to-serial bus conversion can be "programmed" to start in any register by setting the appropriate binary value on the counter preload inputs and applying a pulse to the sync input. If the loading sequence produced by the counter is not required, it can be disabled by connecting the "clock" to "sync" input. At each positive clock edge, the register loaded will depend upon the data at the counter inputs on the previous positive clock edge. The 3-to-8 decode circuit produces load pulses to latch parallel data into the shift registers. Figure 5 shows the parallel-to-serial conversion data matrix.

The description above shows the conversion of parallel data into eight streams of serial data. However, the same circuit also performs serial-to-parallel conversion. A serial eight-bit data stream on one of the eight inputs appears as an eight-bit parallel word on the eight outputs. Successive parallel words appearing at the eight outputs correspond to the serial data on each of the eight inputs in rotation. See Figure 5.

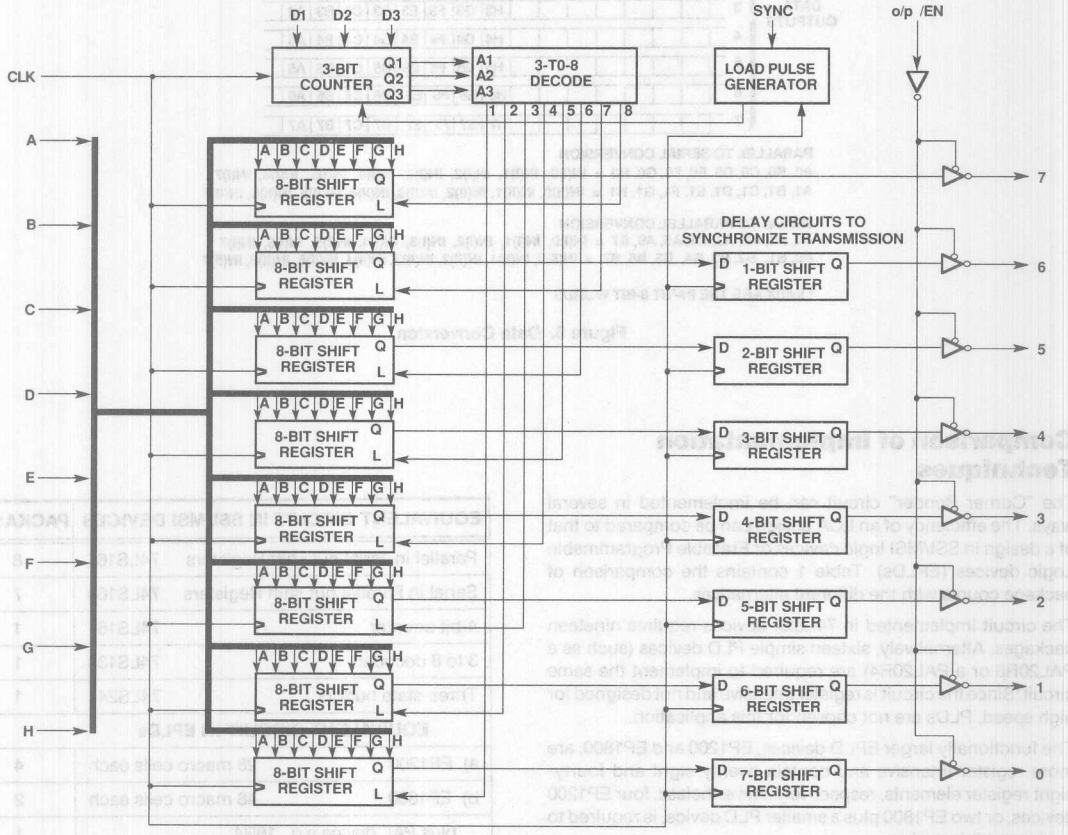
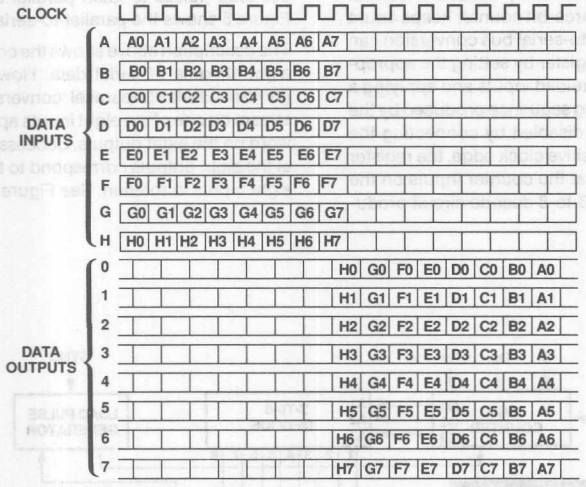


Figure 4. 8-Bit Format Converter ("Corner Bender") Circuit



PARALLEL TO SERIAL CONVERSION
A0, B0, C0, D0, E0, F0, G0, H0 = IN(I)0, IN(I)1, IN(I)2, IN(I)3, IN(I)4, IN(I)5, IN(I)6, IN(I)7
A1, B1, C1, D1, E1, F1, G1, H1 = IN(II)0, IN(II)1, IN(II)2, IN(II)3, IN(II)4, IN(II)5, IN(II)6, IN(II)7

SERIAL TO PARALLEL CONVERSION
A0, A1, A2, A3, A4, A5, A6, A7 = IN(I)0, IN(I)1, IN(I)2, IN(I)3, IN(I)4, IN(I)5, IN(I)6, IN(I)7
B0, B1, B2, B3, B4, B5, B6, B7 = IN(II)0, IN(II)1, IN(II)2, IN(II)3, IN(II)4, IN(II)5, IN(II)6, IN(II)7

* IN(I)X ARE THE INPUT 8-BIT WORDS

Figure 5. Data Conversion

Comparison of Implementation Techniques

The "Corner Bender" circuit can be implemented in several ways. The efficiency of an LCA design can be compared to that of a design in SSI/MSI logic devices or Erasable Programmable Logic devices (EPLDs). Table 1 contains the comparison of package counts with the different alternatives.

The circuit implemented in 74LSxx devices requires nineteen packages. Alternatively, sixteen simple PLD devices (such as a PAL20R8 or a PAL20R4) are required to implement the same circuit. Since the circuit is register intensive and not designed for high speed, PLDs are not chosen for this application.

The functionally larger EPLD devices, EP1200 and EP1800, are more register intensive and contain twenty-eight and fourty-eight register elements, respectively. Nevertheless, four EP1200 devices, or two EP1800 plus a smaller PLD device, is required to implement this design.

The entire Corner Bender circuit fits neatly into a single LCA 2018 device. It uses ninety-nine out of the one hundred available internal macrocells and demonstrates efficient implementation of shift registers and small counters. An LSI device from Plessey (Reference 2) is available to implement the same circuit. However, it is an NMOS device and limited to speeds of under 2 MHz.

EQUIVALENT CIRCUIT IN SSI/MSI DEVICES		PACKAGES
Parallel in serial out shift Registers	74LS165	8
Serial in Parallel out shift Registers	74LS164	7
4-bit counter	74LS161	1
3 to 8 decoder	74LS138	1
Three-state buffers	74LS241	1
EQUIVALENT CIRCUIT IN EPLDs		
a) EP1200	28 macro cells each	4
b) EP1800	48 macro cells each	2
plus PAL device e.g., 16R4		1
EQUIVALENT CIRCUIT IN MMI LCA DEVICE		
LCA 2018 device	100 macro cell	1

Table 1. Implementation Alternatives for the 8-Bit Format Converter Circuit "Corner Bender"

"Corner Bender" Design in an LCA Device

A count of the register elements in the circuit shows a need of sixty-four elements for the eight-shift registers, twenty-eight elements for the delay registers, three elements for the counter, and four elements for the 3-to-8 decode circuit. Each configurable logic block (CLB) in the LCA device can produce two outputs. Hence, only four CLBs are required for the 3-to-8 decode circuit. Since, each CLB contains one register element, the total count of CLBs is therefore ninety-nine. The LCA 2018 contains one hundred CLBs. Twenty-two input/output blocks (IOBs) were used. Figure 6 shows the layout of the design in the LCA device. The circuit fits into a 68-pin PLCC package.

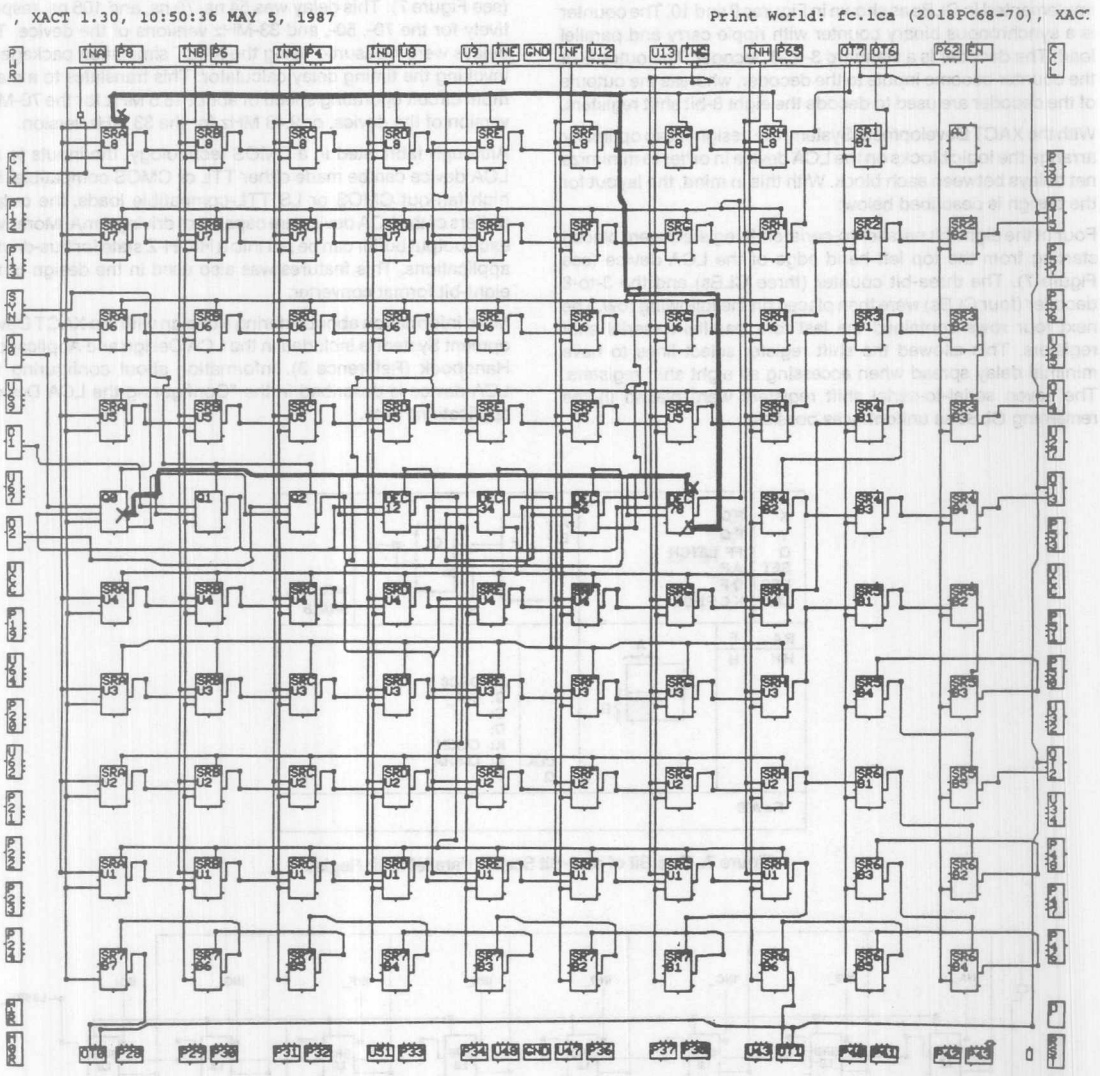


Figure 6. Layout Diagram of "Corner Bender" Circuit on LCA Device of Size 10x10

The Monolithic Memories' XACT™ Design Editor is used to create the design by implementing the appropriate logic functions in CLBs and IOBs. An example of a CLB for one bit of the eight-bit shift register in this design is shown in Figure 7. Input A is the decoder input and it is ANDed with input B, the data to be shifted. To implement the eight-bit shift registers from the one-bit shift register, eight of the one-bit shift registers were linked together so that the output of each register became an input to the next register. This is shown in Figure 8 where each CLB represents one bit of the eight-bit shift register.

The three-bit counter and the 3-to-8 decoder in this design were implemented in CLBs as shown in Figures 9 and 10. The counter is a synchronous binary counter with ripple carry and parallel load. The decoder is a standard 3-to-8 decoder. The outputs of the counter become inputs to the decoder, whereas the outputs of the decoder are used to decode the eight 8-bit shift registers.

With the XACT Development System, the designer can optimally arrange the logic blocks on the LCA device in order to minimize net delays between each block. With this in mind, the layout for the design is described below:

Four of the eight-bit parallel-to-serial shift registers were placed starting from the top left-hand edge of the LCA device (see Figure 7). The three-bit counter (three CLBs) and the 3-to-8 decoder (four CLBs) were then placed on the following row. The next four rows contained the last four parallel-to-serial shift registers. This allowed the shift register select lines to have minimal delay spread when accessing all eight shift registers. The seven serial-to-serial shift registers were placed in the remaining CLBs as uniformly as possible.

The optimum placement and distribution of configured blocks in the array is influenced by the performance needs of the system. Blocks placed in close proximity can use local interconnection resources which incur short signal propagation delays, whereas blocks placed further apart must use either "long lines" or other interconnection resources. Manual optimization using the delay efficient "long lines" was performed for the most critical net connections. After routing completion, the longest delay between two clock pulses was the delay for the counter to change state, the state which is decoded via the 3-to-8 decoder and selects the appropriate shift register to load the parallel data (see Figure 7). This delay was 54 ns, 79 ns, and 106 ns, respectively for the 70-, 50-, and 33-MHz versions of the device. The delays were measured using the XACT simulation package by invoking the timing delay calculator. This translates to a maximum circuit operating speed of about 18.5 MHz for the 70-MHz version of the device, or 9.43 MHz for the 33 MHz version.

Although fabricated in a CMOS technology, the inputs to the LCA device can be made either TTL or CMOS compatible. For high fan-out CMOS or LS TTL-compatible loads, the output buffers of the LCA device are capable of driving 4 mA. Moreover, each output buffer can be put into a HIGH-Z state for bus-driving applications. This features was also used in the design of the eight-bit format converter.

More information about entering a design with the XACT Development System is included in the LCA Design and Applications Handbook (Reference 3). Information about configuring the LCA device is described in the "Configuring the LCA Device" applications note.

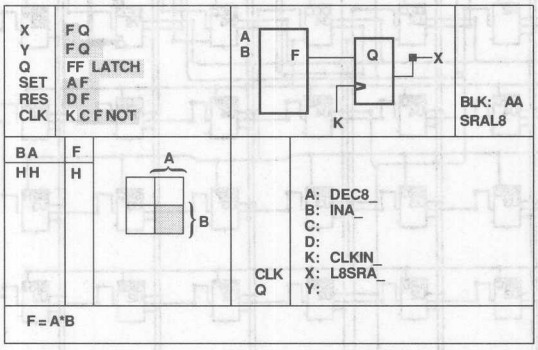


Figure 7. One Bit of an 8-Bit Serial - Parallel Shift Register

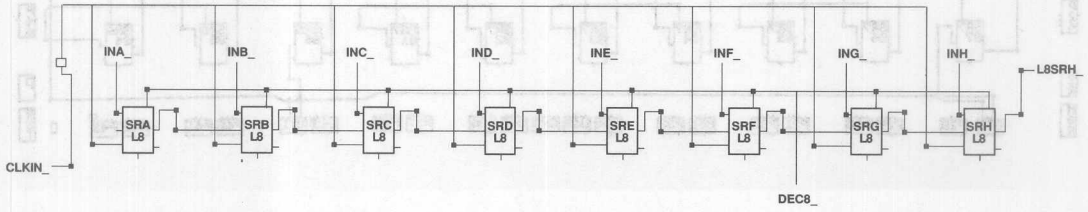


Figure 8. CLB Schematic Output for the 8-Bit Shift Register

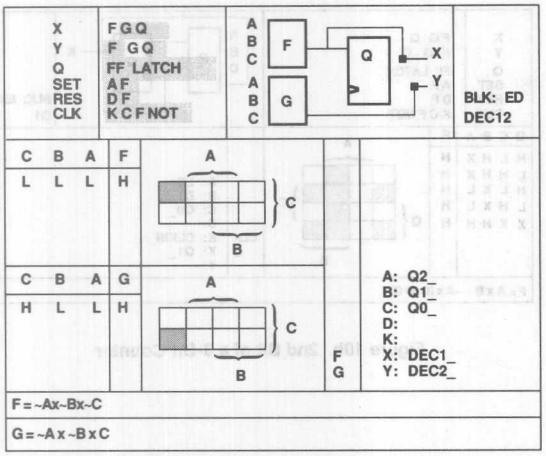


Figure 9a. Decode Outputs 1 and 2 of 3-to-8 Decoder

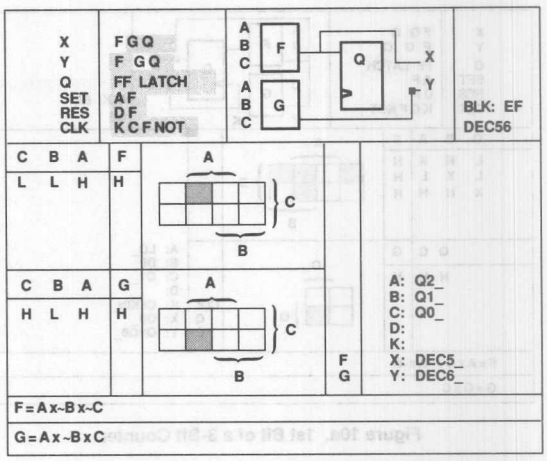


Figure 9c. Decode Outputs 5 and 6 of 3-to-8 Decoder

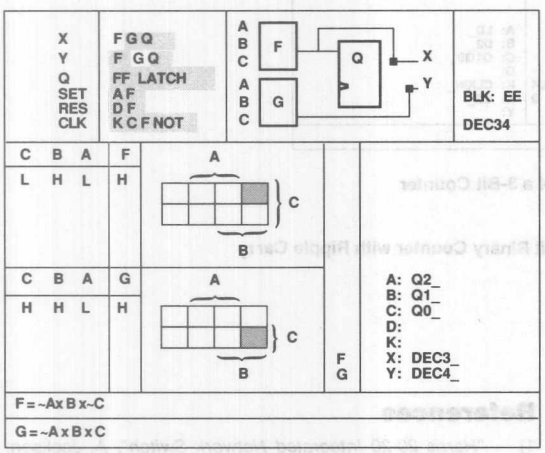


Figure 9b. Decode Outputs 3 and 4 of 3-to-6 Decoder

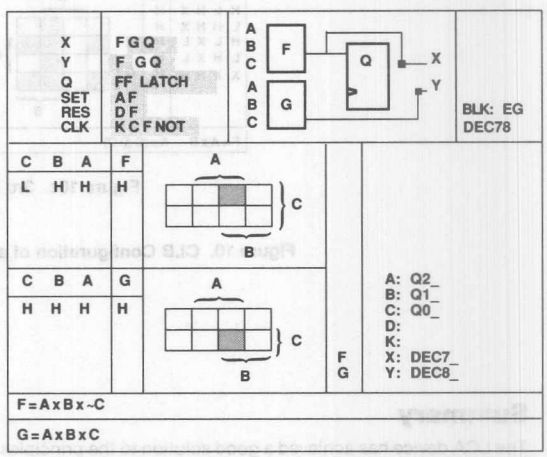


Figure 9d. Decode Outputs 7 and 8 of 3-to-8 Decoder

Figure 9. CLB Configuration of a 3-to-8 Decoder

5

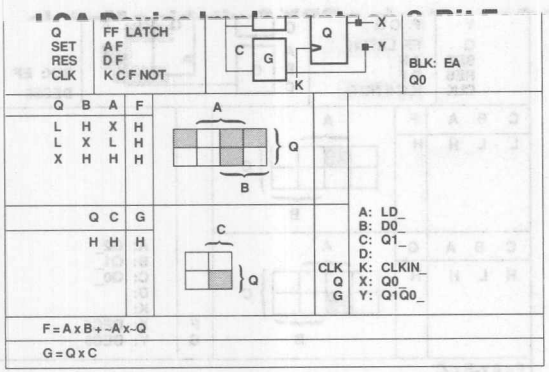


Figure 10a. 1st Bit of a 3-Bit Counter

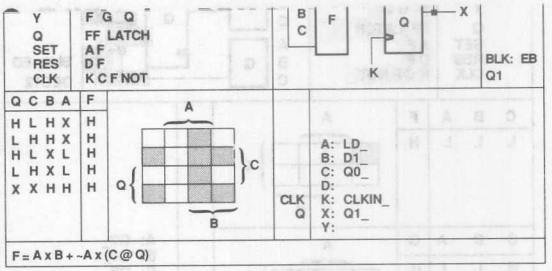


Figure 10b. 2nd Bit of a 3-Bit Counter

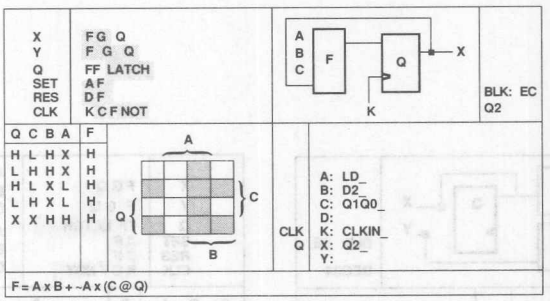


Figure 10c. 3rd Bit of a 3-Bit Counter

Figure 10. CLB Configuration of a 3-Bit Binary Counter with Ripple Carry

Summary

The LCA device has achieved a good solution to the principles used for multiplexing digitally-encoded voice channels in both serial and parallel hierarchies. These hierarchies can accommodate thousands of voice circuits in a PBX switching module. The detailed design of the eight-bit (parallel-to-serial and serial-to-parallel) format converter circuit, or Corner Bender, developed in Monolithic Memories' LCA M2018 device was shown to be efficiently implemented. Despite the fast development of the system using the XACT Design Editor, the final design was programmed and bench tested to verify functional integrity.

The Corner Bender design is available from Monolithic Memories upon request. The bit pattern and .LCA file will be provided for programming the LCA device in an EPROM. Please ask for design XDES05.LCA

References

1) *"Harris 20-20 Integrated Network Switch"*, A. Jackson, IEEE SAC-3, No. 4 July 1985, p561-568.

2) MJ1410, 8-bit Format Converter, Data Sheet, Plessey Semiconductors.

3) Logic Cell Array Design and Applications Handbook, Monolithic Memories, 1987.

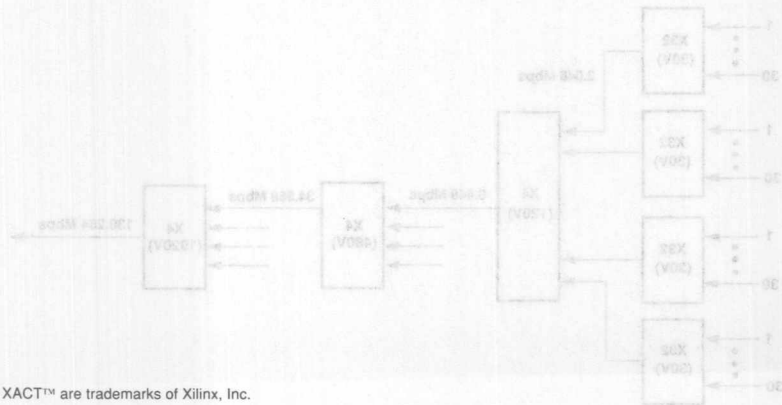
Reconfigurable Programmable Devices (LCA) Simplify Digital TDM Line Transcoder

C. B. Lee and Cindy Lee

Abstract

The Logic Cell™ Array (LCA) is a high-density CMOS programmable integrated circuit. Its reprogrammability and reconfigurability make complex design of custom LSI devices easy and flexible.

This application note describes the design of a universal transcoder on trunk transmission lines. Any one of three different transcoders (B8ZS, B6ZS, and HDB3) can be implemented in a single LCA device (M2018). Because it is reconfigurable, any one of these different transcoders may be selected. Other common communication features can be implemented simply by reconfiguring the Configurable Logic Blocks (CLBs), Input/Output Blocks (IOBs), or interconnect.



TWX: 910-338-2376
2175 Mission College Blvd. Santa Clara, CA 95054-1592 Tel: (408) 970-9700 TWX: 910-338-2374
Monolithic Memories

Reconfigurable Programmable Devices (LCA) Simplify Digital TDM Line Transcoder

C. B. Lee and Cindy Lee

TDM Hierarchies

Recently, the International Telegraph and Telephone Consultative Committee (CCITT) agreed on transmission hierarchies based on different national standards. Two major hierarchies are in use today. The first hierarchy, known as the T-carrier, is used in North America and Japan and multiplexes twenty-four voice channels together using Time Division Multiplexing (TDM) technique. T-carrier is based on a primary rate of 1.544 Mbps and multiples up to 274.176 Mbps (see Figure1).

The other hierarchy, used in Europe, Africa, and South America, multiplexes thirty voice channels and two signaling channels together. This TDM hierarchy is based on a primary rate of 2.048 Mbps and multiples up to 139.264 Mbps (see Figure 2) .

CCITT recommendation G.703 defines line coding for both TDM hierarchies with different orders. Table 1 shows a 1.544 Mbps primary rate T-carrier hierarchy with four different orders. Alternate Mark Inversion (AMI) or Binary 8 Zeros Substitution (B8ZS) is the proper line coding in the first order. In the second order, if the transmission medium is coaxial pair cables, then B8ZS is used. Otherwise, Binary 6 Zeros Substitution (B6ZS) is used for symmetrical pair cables. In the third order, Binary 3 Zeros Substitution (B3ZS) is used, while polar binary NRZ is used in the fourth order.

Similarly, Table 2 shows line coding based on a 2.048 Mbps primary rate with four different orders. High Density Bipolar (HDB3) is used in the first, second, and third orders. Finally, Code Mark Inversion (CMI) is used in the fourth order.

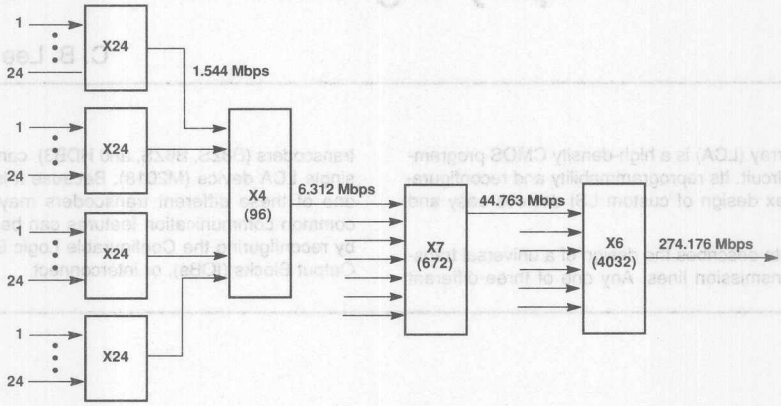


Figure 1. TDM Hierarchy at 1.544 Mbps Primary Rate

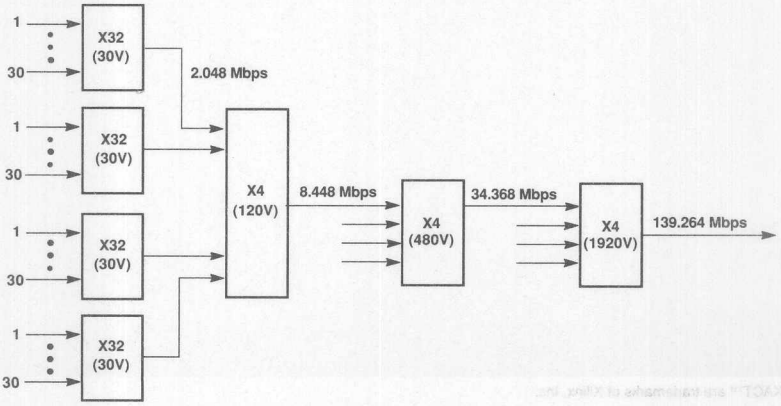


Figure 2. TDM Hierarchy at 2.048 Mbps Primary Rate

ORDER	CHANNEL BANK	NUMBER OF VOICE CHANNELS	BIT RATE (Mbps)	LINE CODING
1	DS-1	24	1.544	AMI or B8ZS
2	DS-2	96	6.312	B8ZS-Coaxial Pair B6ZS-Symmetric Pair
3	DS-3	672	44.736	B3ZS
4	DS-4	4032	274.176	Polar Binary

Table 1. Line Coding for TDM Hierarchy at 1.544 Mbps Primary Rate

ORDER	NUMBER OF VOICE CHANNELS	BIT RATE (Mbps)	LINE CODING
1	30	2.048	HDB3
2	120	8.448	HDB3
3	480	34.368	HDB3
4	1920	139.264	CMI (2-level NRZ)

Table 2. Line Coding for TDM Hierarchy at 2.048 Mbps Primary Rate

Binary Codes

Unipolar binary codes form the information exchange in computer systems from one device to another over short distances. Unipolar binary codes work well at short distances but DC wander and a lack of timing information make them unsuitable for long transmission lines. DC wander is caused from unipolar binary signals being transmitted via a certain media over long-distance transmission lines. The signals will be attenuated after a few kilometers.

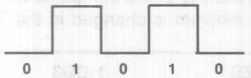


Figure 3a. Unipolar Binary Codes

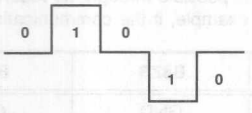


Figure 3b. Bipolar Codes

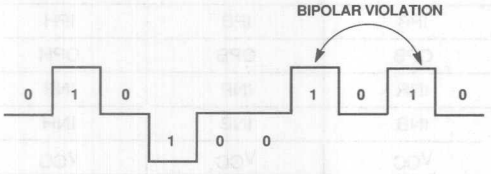


Figure 4. Bipolar Violation

Bipolar Codes

Unipolar binary codes require two voltages to represent a binary state (see Figure 3a). On the other hand, bipolar binary codes, also known as AMI codes, need three different voltages (+5 V, 0 V, and -5 V) to represent the same binary condition: logic zero or logic one (see Figure 3b). By making logic one signals alternate between +5 V and -5 V, DC wander is eliminated on a long transmission line because the mean DC level is integrated to zero volts. If, however, two continuous ones have the same polarity a bipolar violation occurs (see Figure 4). A long string of ones of bipolar codes can provide timing information but a long string of zeros cannot. To counteract this problem, several modified bipolar codes provide timing information for a long string of logic zeros.

Modified Bipolar Codes

The B8ZS, B6ZS, and HDB3 codes are all modified bipolar codes. They provide timing information for a long string of zeros by forcing bipolar violations.

B8ZS line coding encodes logic ones as alternating positive (+5 V) and negative (-5 V) signals. It substitutes any continuous sequence of eight zeros with a special pattern (see Table 3). Depending on the preceding polarity, one of two B8ZS substitution codes is generated: 000+-0-+ or 000-+0+-. The "+" indicates positive voltage, "-" indicates negative voltage, and "0" indicates zero voltage. If the preceding polarity is positive, 000+-0-+ is generated; otherwise, 000-+0+- is generated. The B8ZS substitution code forces bipolar violations in the fourth and the seventh bit positions.

B6ZS is similar to B8ZS. Rather than detecting eight continuous zeros, B6ZS substitutes for six continuous zeros with a special pattern. The B6ZS substitution codes are 0+-0-+ and 0-+0+-, depending on the preceding polarity. B6ZS substitution code forces the second and fifth zeros to bipolar violation (see Table 4).

PRECEDING POLARITY	B8ZS SUBSTITUTION
+	000+-0-+
-	000-+0+-

Table 3. B8ZS Substitution Table

PRECEDING POLARITY	B6ZS SUBSTITUTION
+	0+-0-+
-	0-+0+-

Table 4. B6ZS Substitution Table

tutes any four continuous zeros with a special HDB3 substitution code. This HDB3 substitution code forces the fourth bit to be a bipolar violation. The four possible HDB3 substitution codes are: +00+, 000+, -00-, and 000-. The choice of substitution code depends on the preceding polarity and the number of logic ones after the preceding bipolar violation. When the number of logic ones is odd, 000V is generated. Otherwise, B00V is generated. "V" indicates the bipolar violation while "B" indicates the bipolar signal. The polarity of V in 000V is the same as that of the preceding logic one signal. If the HDB3 code is B00V, the polarity of B is opposite that of the preceding logic one signal, and the polarity of V is the same as B (see Table 5).

PRECEDING POLARITY	NUMBER OF LOGIC ONES AFTER THE PREVIOUS BIPOLAR VIOLATION	
	ODD	EVEN
Positive +	000+	-00-
Negative -	000-	+00+

Table 5. HDB3 Substitution Table

Logic Cell Array Implementation

The Logic Cell Array (LCA) device is a high-density CMOS integrated circuit. Its user-programmable static RAM array architecture consists of Input/Output Blocks (IOBs), Configurable Logic Blocks (CLBs), and Interconnect. All of them are fully user-programmable and user-reconfigurable. High-density, programmability and reconfigurability of LCA devices allow

Currently, two LCA devices are available, their complexity is based on the number of CLBs within the device. All CLBs are arranged in a matrix in the center of the device. The M2064 has sixty-four CLBs which are arranged in an 8-row by 8-column matrix. The M2018 has one hundred CLBs arranged as a 10 by 10 matrix.

A CLB can be configured to be a storage element (registered or latched), a combinational logic block or a mixed combinational logic block with storage element. Each CLB has four general-purpose inputs (A, B, C, and D); and a special clock input (K). Also, each CLB can perform any function of up to four variables or any two functions of up to three variables. These variables can be selected from external inputs or from the internal register feedback path.

Initially, an attempt was made to design the line transcoder with the M2064 device. However, all three line transcoder designs required more than sixty-four CLBs each. Fortunately, it is easy to convert a design from M2064 to M2018 because the change is made just by selecting the M2018 part type in the convert command when using the XACT Design Editor. Any of the preceding line coding schemes can be implemented in a single M2018 because of its higher density (100 CLBs). Since the LCA device is reconfigurable, a line transcoder implemented in a single LCA device (M2018),can support the North American T1 carrier (B8ZS line coding) or T2 carrier (B6ZS line coding), or even the European standard (HDB3 line coding).

All three line transcoders can be configured and have the same pinout assignments. This offers a simple solution for supporting the different standards without component replacement. Once a design has been established, "on-the-fly" modification is possible through the reconfigurability of the component. For example, if the communication medium is changed in the T2

PIN NUMBER	DESCRIPTION	B8ZS	B6ZS	HDB3
Pin 01	GND	GND	GND	GND
Pin 11	Clock	CLK	CLK	CLK
Pin 12	Master Reset	MR	MR	MR
Pin 13	Positive input for decoder	IPB	IPB	IPH
Pin 14	Positive output from encoder	OPB	OPB	OPH
Pin 15	NRZ input	INR	INR	INR
Pin 17	Negative input for decoder	INB	INB	INH
Pin 18	V_{CC}	V_{CC}	V_{CC}	V_{CC}
Pin 19	Negative output from encoder	ONB	ONB	ONH
Pin 34	NRZ output	ONR	ONR	ONR
Pin 35	GND	GND	GND	GND
Pin 52	V_{CC}	V_{CC}	V_{CC}	V_{CC}
Pin 56	Error*	ERR	ERR	ERR
Pin 57	Bipolar violation error	BVP	BVP	BVP
Pin 59	Special pattern detected	B8Z	B6Z	HDB

* Means that the ERROR is asserted when both positive and negative signals are high simultaneously.

Table 6. Cross Reference of Pin Names for Three Line Transcoders

standard, the transcoder may be reprogrammed from B8ZS to B6ZS, or vice versa, without any other alterations to the hardware. In total, fifteen pins are used in these designs; four power pins, two central control pins, three pins for the encoder, and six pins for the decoder. Table 6 cross references the pin names for the three different line transcoders.

B8ZS Line Coding

A B8ZS encoder converts NRZ data to B8ZS code. The B8ZS encoder consists of a 3-CLB, 3-bit counter, a 5-CLB, 5-bit shift register, a 6-CLB sequence state machine, and a 13-CLB encoding state machine (see Figure 5). A 3-bit counter detects eight continuous zeros. A 5-bit shift register delays the NRZ input data to synchronize with the 3-bit counter's outputs. The sequence state machine provides the sequence state (Q2, Q1, and Q0) for encoding state machine. The encoding state machine generates the B8ZS code. It is a pair of signals which interface to line driver to provide three level signals: positive (PB8ZS), negative (NB8ZS), and zero signals.

A B8ZS decoder detects B8ZS code on a pair of positive and negative B8ZS signals (see Figure 6). When either of the following sequences: 000+-0-+ or 000-+0+- is detected, the decoder generates a string of eight zeros. Otherwise, the encoder converts bipolar signals to binary NRZ data.

Figure 7 is a block diagram of the B8ZS decoder. It includes a 19-CLB decoding state machine, an 8-CLB, 8-bit shift register, a 5-CLB RESET generator, and a 7-CLB bipolar violation and error flag generator. Four state variables (HQ3, HQ2, HQ1, and HQ0) are generated from the decoding state machine. The 8-bit shift register buffers DSOUT data in order to provide eight continuous zeros when the B8ZS signal is asserted. The B8ZS signal is generated when the B8ZS substitution code is detected. The bipolar violation error (BVP) occurs when any two sequential signals of the same polarity are detected except the B8ZS substitution code. If both positive B8ZS and negative B8ZS are detected simultaneously, the error signal (ERROR) is generated.

The implementation details of the B8ZS line transcoder within a single M2018 are shown in Figure-8. This design requires sixty-six out of the one hundred available CLBs.

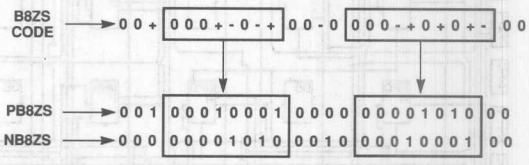


Figure 6. B8ZS Codes in Positive-B8ZS and Negative-B8ZS Signals

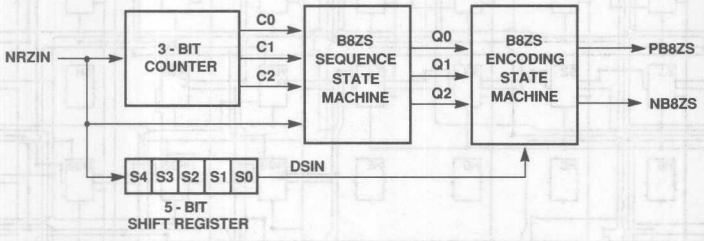


Figure 5. Block Diagram of the B8ZS Encoder

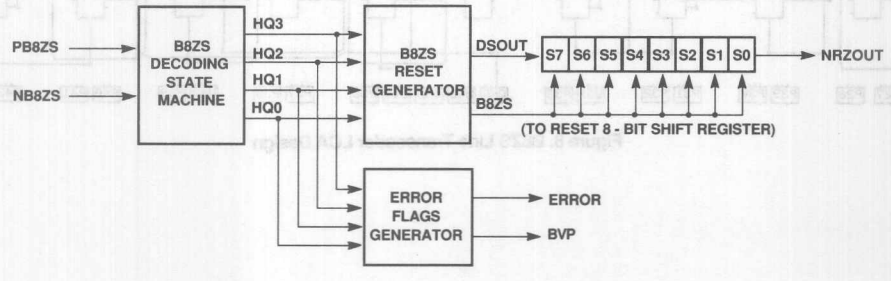


Figure 7. Block Diagram of the B8ZS Decoder

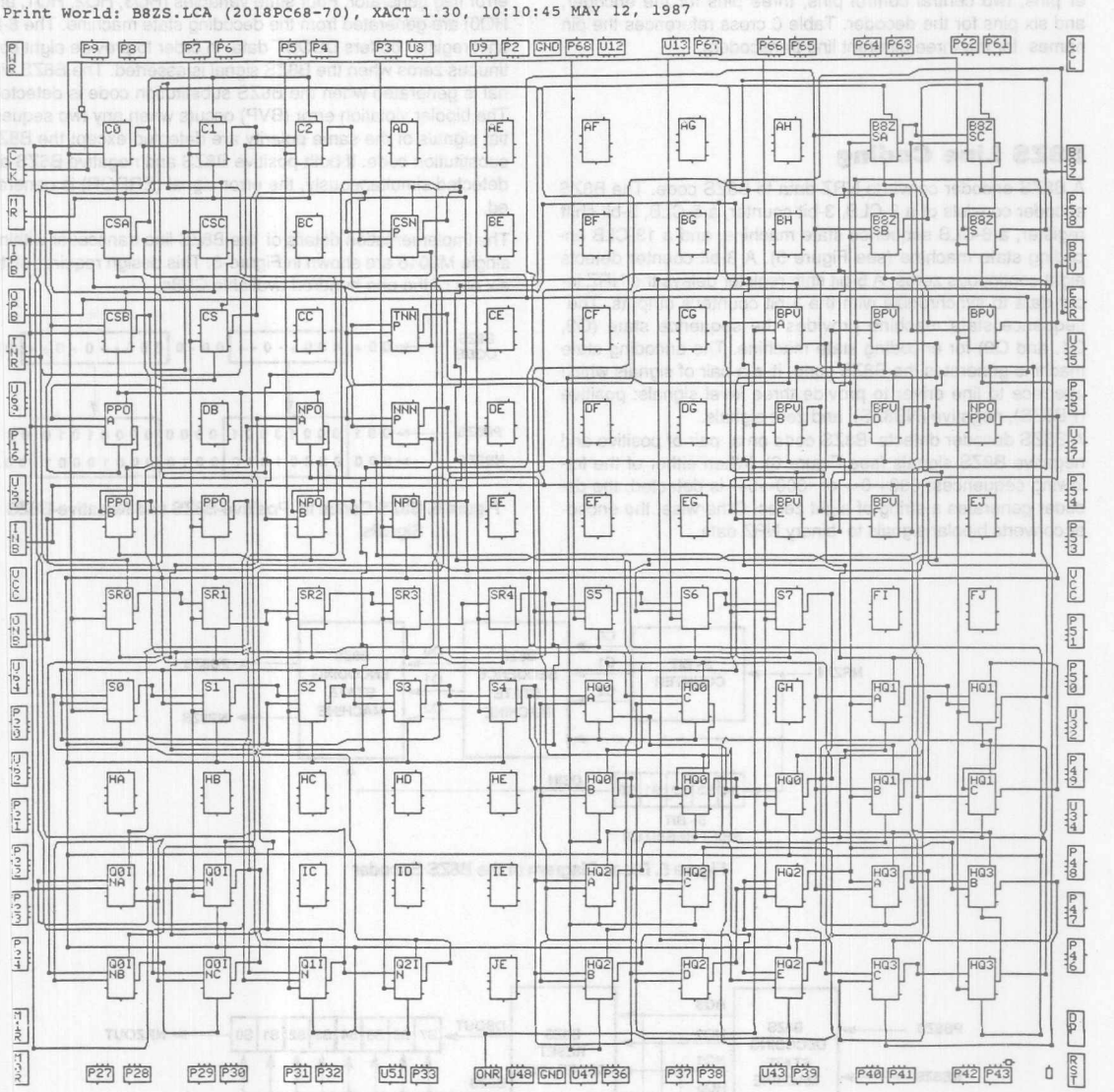


Figure 8. B8ZS Line Transcoder LCA Design

B6ZS Line Coding

The M2018 implementation of the B6ZS line transcoder is shown in Figure 9. In total, sixty-seven CLBs are used to create the design. The single difference between a B8ZS encoder and a B6ZS encoder is the count for continuous zeros. While the 3-bit counter counts eight continuous zeros for the B8ZS encoder, it counts six for the B6ZS encoder. The B8ZS encoder consists of a 5-CLB, six-zeros detector, a 5-CLB, 5-bit shift register, a 6-CLB sequence state machine, and a 13-CLB encoding state machine.

A B6ZS decoder detects two B6ZS substitution codes: 0+-0-+ or 0-+0+-. When one of those two B6ZS substitution codes is detected, the decoder generates six continuous zeros. Otherwise, the decoder converts bipolar signals to binary NRZ data. The B6ZS decoder includes a 20-CLB decoding state machine, a 6-CLB, 6-bit shift register, a 5-CLB RESET generator, and a 7-CLB bipolar violation and error flag generator. The 6-bit shift register is needed to buffer DSOUT data in order to provide six continuous zeros.

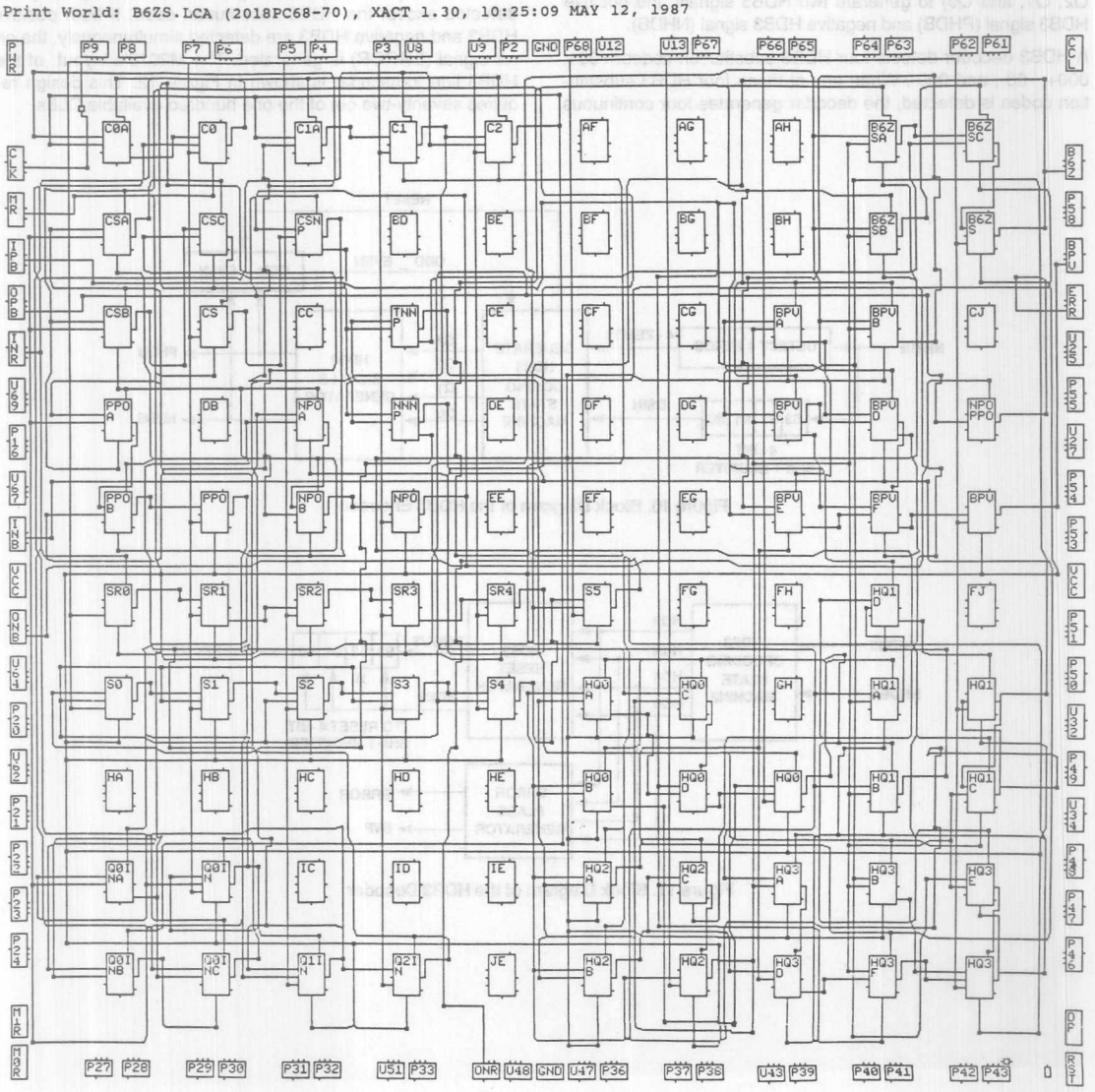


Figure 9. B6ZS Line Transcoder LCA Design

HDB3 Line Coding

In Figure 10, the HDB3 encoder includes a 4-CLB, 4-bit shift register, a 5-CLB HDB3 signal generator, and three-state machines. These state machines are, a 3-CLB ODD_EVEN state machine, a 5-CLB DETECT_4_ZEROS state machine, and 23-CLB encoding state machine. The ODD_EVEN state machine detects the polarity of the preceding pulse. The DETECT_4_ZEROS state machine detects four continuous zeros. The 4-bit shift register delays NRZIN data to synchronize with the output of DETECT_4_ZEROS state machine. The encoding state machine provides four encoding states variables (Q3, Q2, Q1, and Q0) to generate two HDB3 signals: the positive HDB3 signal (PHDB) and negative HDB3 signal (NHDB).

A HDB3 decoder detects four HDB3 substitution codes: +00+, 000+, -00-, and 000-. When any of these four HDB3 substitution codes is detected, the decoder generates four continuous zeros. Otherwise, the decoder converts bipolar signals to binary NRZ data. Figure 11 is a block diagram of the HDB3 decoder. It includes a 23-CLB decoding state machine, a 4-CLB, 4-bit shift register, a 3-CLB RESET generator, and a 2-CLB bipolar violation and error flags generator. Four state variables (HQ3, HQ2, HQ1, and HQ0) are generated from the decoding state machine. The 4-bit shift register buffers DSOUT data in order to provide four continuous zeros when the HDB3 signal is asserted. The HDB3 signal is generated when the HDB3 substitution code is detected. The bipolar violation error (BVP) occurs when any two sequential signals of the same polarity are detected except the HDB3 substitution code. If both positive HDB3 and negative HDB3 are detected simultaneously, the error signal (ERROR) is generated. The M2018's layout of the HDB3 line transcoder is shown in Figure 12. This design requires seventy-two out of the one hundred available CLBs.

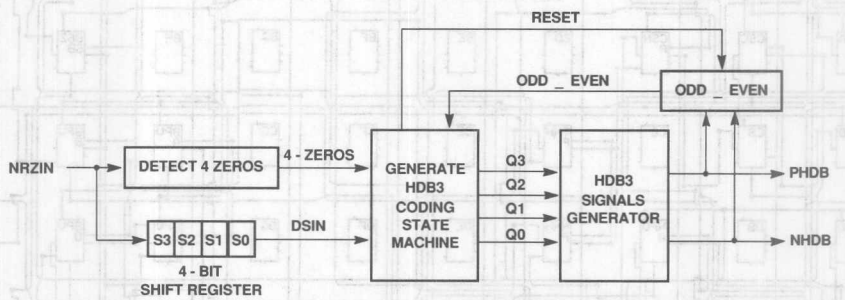


Figure 10. Block Diagram of the HDB3 Encoder

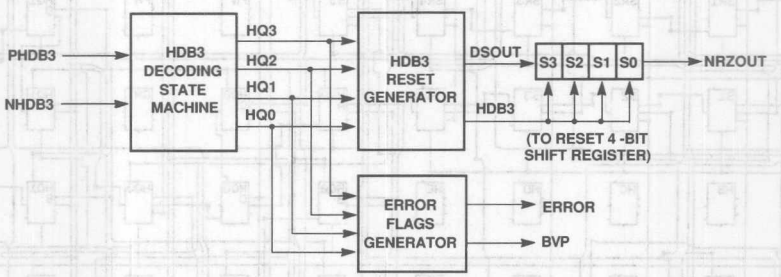


Figure 11. Block Diagram of the HDB3 Decoder

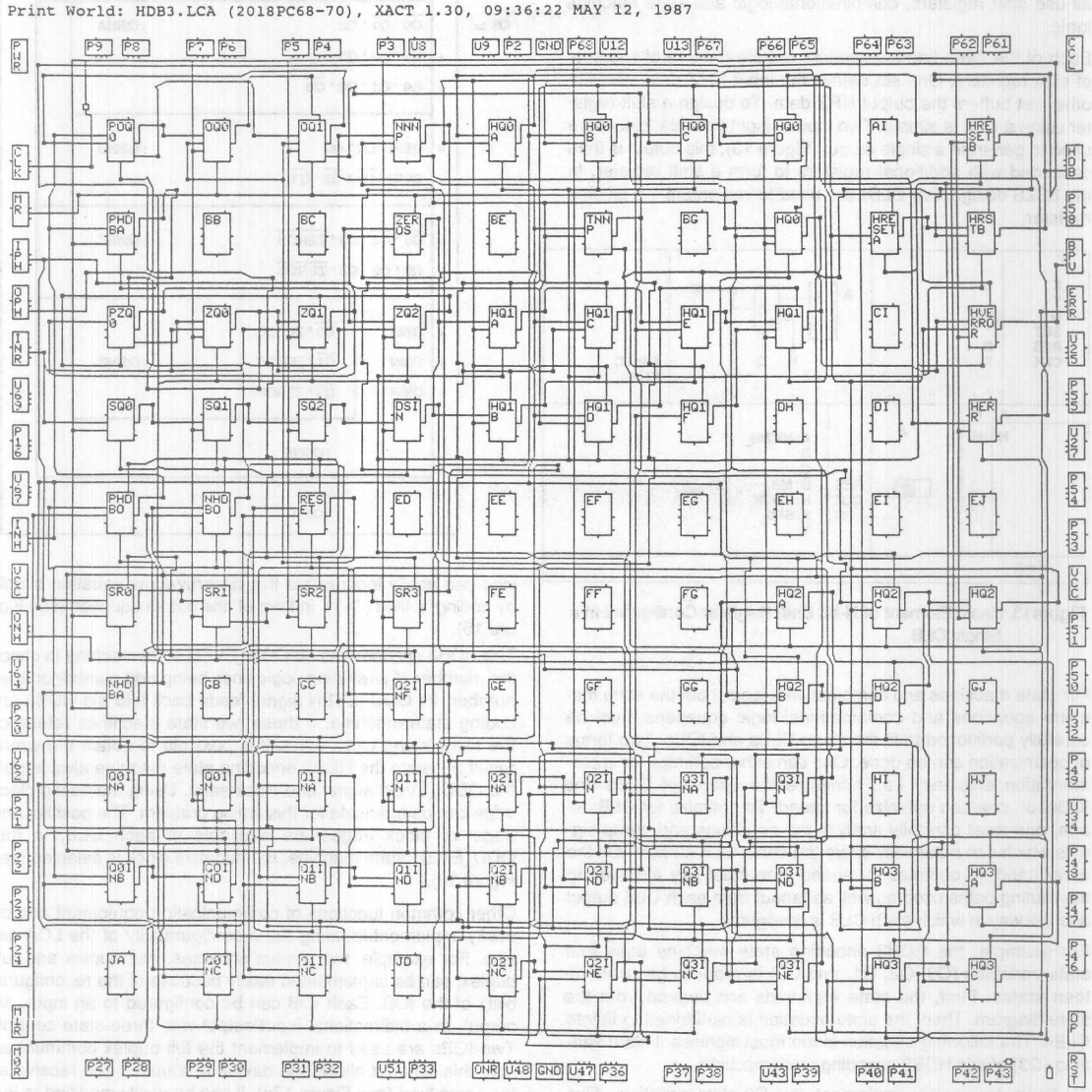


Figure 12. HDB3 Line Transcoder LCA Design

Logic Cell Array Design Methodology

All three designs are composed of similar building blocks, they all use shift registers, combinational logic and state machine logic.

Each of the three line transcoders require the use of two sets of shift registers. One set delays the input NRZ data, and the other set buffers the output NRZ data. To design a shift register using a CLB is simple. Two inputs (input variable, clock) are used to generate a single output (Figure 13); this output is then cascaded with additional registers to form a shift register. In the B8ZS design, five CLBs are used to implement a 5-bit shift register.

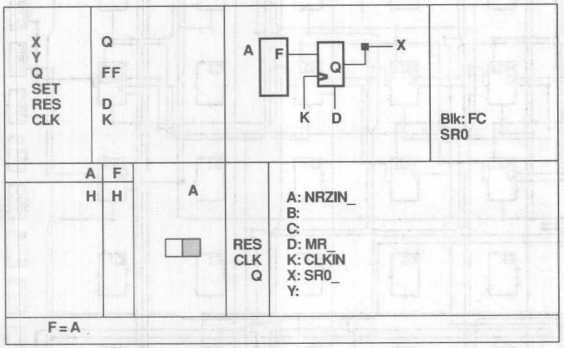


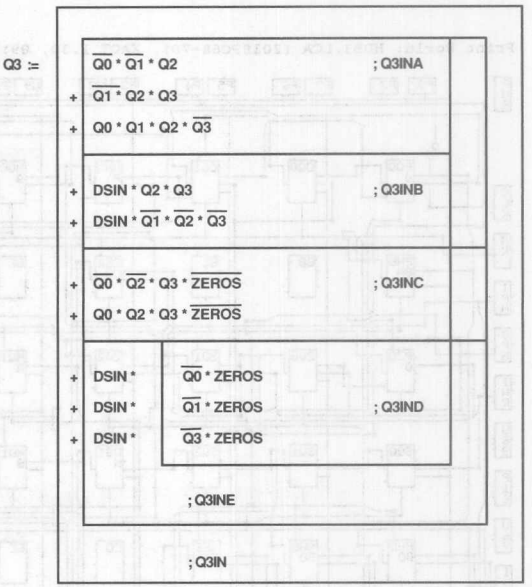
Figure13. Basic Element of N-bit Shift Register Configured in a Single CLB

For state machines and combinational logic, both the state machine equations and combinational logic equations must be carefully partitioned to fit them into CLBs and IOBs. Two forms of optimization can be done. One can either optimize for implementation efficiency i.e., minimize the usage of CLBs and IOBs, or, one can optimize for speed. To optimize for CLB usage, one must carefully analyze the equations and group signals which can potentially share the same CLB outputs. On the other hand, to optimize for speed, one must pay attention to the routing paths used as well as fanout from each CLB output and the way in which each CLB is configured.

For example, the HDB3 encoding state machine uses four state variables (Q3, Q2, Q1, and Q0) in order to generate fifteen states. First, the state equations are derived from the state diagram. Then, the state equation is partitioned to fit into CLBs. The following equation is the most significant state variable (Q3) of this HDB3 encoding state machine.

Six CLBs are used to implement this Q3 state equation. Five of the CLBs (Q3INA, Q3INB, Q3INC, Q3IND, and Q3INE) are configured as 4-input variable CLBs which generate a single output. Because these CLBs perform the preliminary logic for input into the block Q3IN, no storage element is necessary in these CLBs. Instead, the storage function is implemented in the block Q3IN. The design details for each CLB is shown in Figure 14.

The B8ZS line transcoder can be implemented using three PAL® devices: two 16R8 devices and one 16R6 device. All equations for those PAL devices are assertive low; e.g., $\overline{C2}$:= C2 * C1 * C0 + $\overline{C2}$ * $\overline{C1}$ + $\overline{C2}$ * $\overline{C0}$ + RST + NRZIN. The LCA device can easily incorporate this assertive low equation simply by adding a tilde ("~") in front of the output equation (see Figure 15).



The HDB3 encoder has an ODD_EVEN state machine to check the number of preceding logic ones being odd number or even number. Its ODD_EVEN signal feeds back into the HDB3 encoding state machine. If these two state machines reference the same rising clock edge. It is possible to obtain the wrong result because the HDB3 encoding state machine always gets the ODD_EVEN signal one clock early. Using the falling clock edge can compensate for this timing problem. The positive and negative clock edges are available in each CLB. In this ODD_EVEN state machine, the negative clock is selected (see Figure 16).

Other common functions of communication equipment can be easily implemented using the reconfigurability of the LCA device. For example, two duplex schemes: half duplex and full duplex, can be implemented easily because of the reconfigurablilty of the IOB. Each IOB can be configured to an input, an output, or a bidirectional input/output with three-state control. Two IOBs are used to implement the full duplex communication, this scheme allows the device to transmit and receive at the same time (see Figure 17a). It can be easily modified to the half-duplex communication simply by using one bidirectional IOB, the information must be transmitted or received exclusively (see Figure 17b).

In addition, loop-back feature can be implemented by reconfiguring LCA's interconnect. Remote loop-back is implemented by disconnecting the line transcoder with the input and output pads, but connecting the input pad to the output pad directly within the transcoder (see Figure 18a). Local loop-back can also be implemented. Connecting the output of the transmitter to the input of the receiver directly, but disconnecting with input and output pads (see Figure 18b).

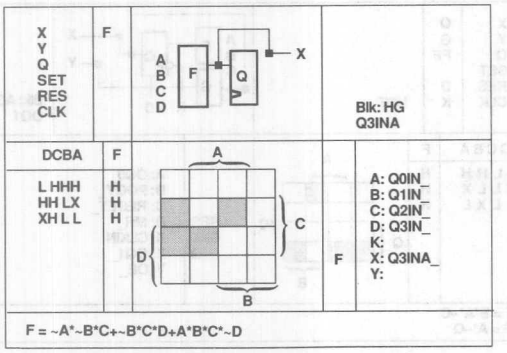


Figure 14a. Q3INA's CLB Design

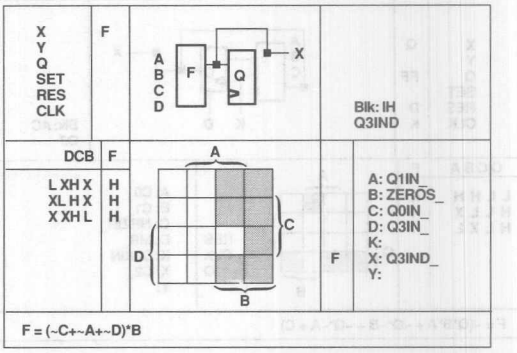


Figure 14d. Q3IND's CLB Design

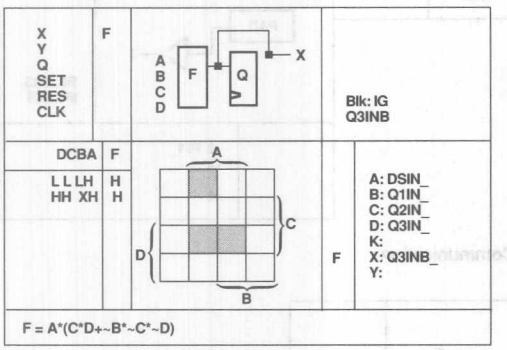


Figure 14b. Q3INB's CLB Design

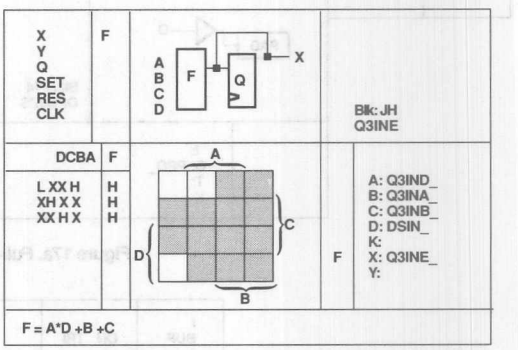


Figure 14e. Q3INE's CLB Design

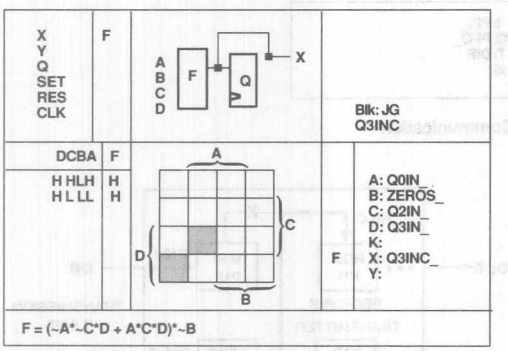


Figure 14c. Q3INC's CLB Design

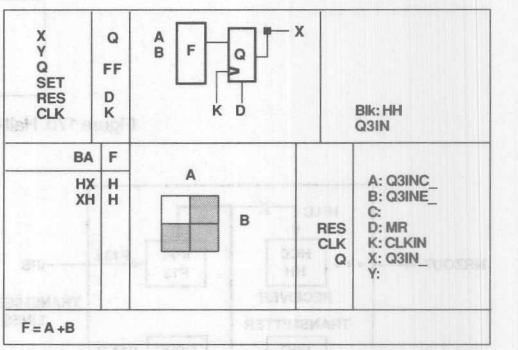


Figure 14f. Q3IN's CLB Design

Figure 14. The CLB's Design for the Q3 Equation

5

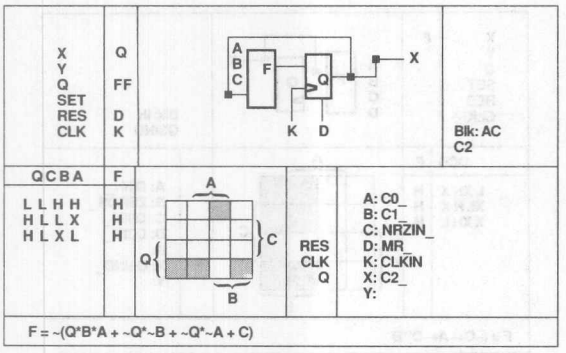


Figure-15. C2 is Represented by ~(C2)

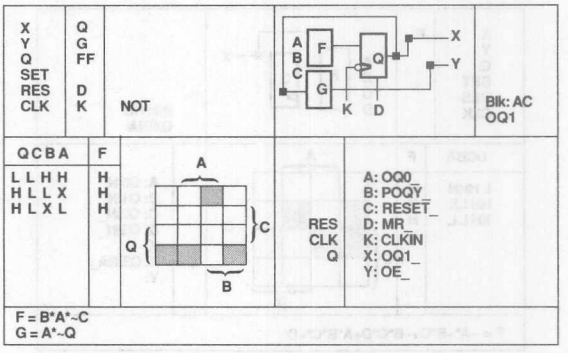


Figure 16. Negative Clock Edge in the CLB

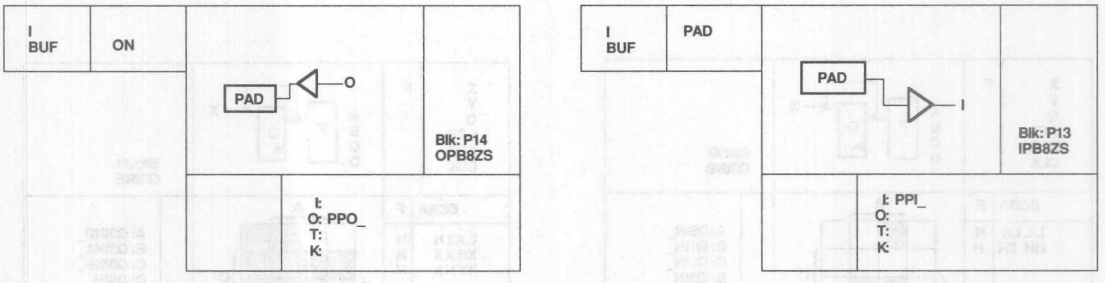


Figure 17a. Full-Duplex Communication

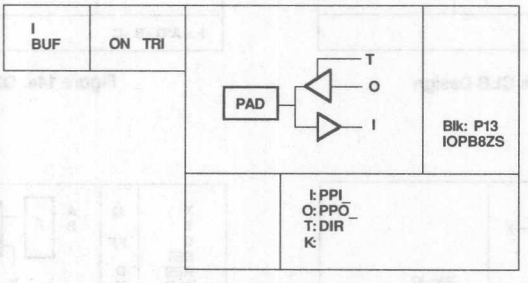


Figure 17b. Half-Duplex Communication

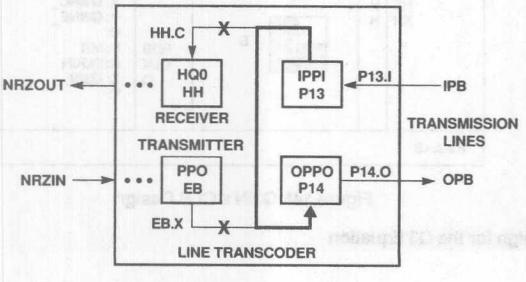


Figure 18a. Remote Loop Back

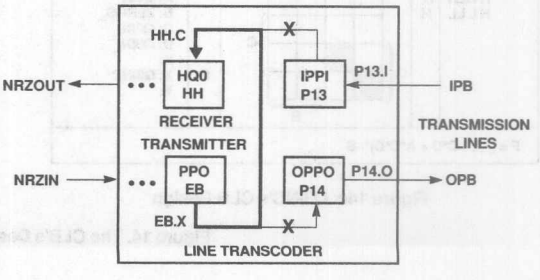


Figure 18b. Local Loop Back

Conclusion

This design demonstrates the power versatility and benefit of using the LCA device in line coding by exploiting the reconfigurable feature of the device. A single M2018 LCA device can support three different line codes, such as B8ZS, B6ZS, and HDB3 codes.

The B8ZS is used at either 1.544 Mbps T1 carrier or 6.312 Mbps T2 carrier on coaxial pairs. The B8ZS encoder uses twenty-seven CLBs to convert NRZ data into two B8ZS signals, and the B8ZS decoder uses thirty-nine CLBs to convert two B8ZS signals to NRZ data. The B6ZS is used at 6.312 Mbps T2 carrier on symmetrical pairs. The B6ZS encoder uses twenty-nine CLBs to convert NRZ data into two B6ZS signals, and the B6ZS decoder uses thirty-eight CLBs to convert two B6ZS signals to NRZ data. The HDB3 is used at 2.048 Mbps or 8.448 Mbps transmission lines. The HDB3 encoder uses forty CLBs to convert NRZ data into two HDB3 signals and the HDB3 decoder uses thirty-two CLBs to convert two HDB3 signals to NRZ data.

The line transcoder is required when the signal is transmitted over transmission lines. It converts unipolar binary codes to bipolar codes or modified bipolar codes in order to eliminate DC wander and provide the timing information.

References

1. Theresa Shafer and Steve Patterson, *"B8ZS Coding,"* Monolithic Memories Application Note AN-169.
2. Cindy Lee, *"HDB3 Line Coding using Three PAL® Devices,"* Monolithic Memories Application Note AN-176.
3. John C. Bellamy, *"Digital Telephony,"* John Wiley & Son, 1982.
4. Recommendation G.703, CCITT Red Book, Volume III.

5

The detailed LCA design files are available from Monolithic Memories, Inc.

The design file of B8ZS refers to XDES12.LCA.

The design file of B6ZS refers to XDES13.LCA.

The design file of HDB3 refers to XDES14.LCA.

Building an ESDI Translator Using the M2064 Logic Cell™ Array

Ken Won and Ken Tseng

Abstract

The ESDI (Enhanced Small Device Interface) Standard is a low-cost, high-performance Winchester disk drive interface. The ESDI Translator is an interface controller that is implemented in an ESDI-compatible disk drive.

The ESDI Translator is implemented in one M2064 LCA device. The LCA device is a RAM-based high-density CMOS integrated circuit which is reprogrammable. This application note describes the design implementation and design considerations of the ESDI Translator on the LCA device.

TWX: 910-338-2376
2175 Mission College Blvd. Santa Clara, CA 95054-1592 Tel: (408) 970-9700 TWX: 910-338-2374
Monolithic Memories

Building an ESDI Translator Using the M2064 Logic Cell Array

by Ken Won and Ken Tseng

The ESDI Translator

ESDI is a low-cost, high-performance interface standard suitable for the smaller, high-performance Winchester disk drives currently being produced. The ESDI interface consists of a control cable and a data cable. The control cable allows for a daisy chain connection of up to seven devices (disk or optical drives) with only the last device being terminated. In our design, we assumed that the device is a disk drive. The data cable is attached in a radial configuration (See Figure 1).

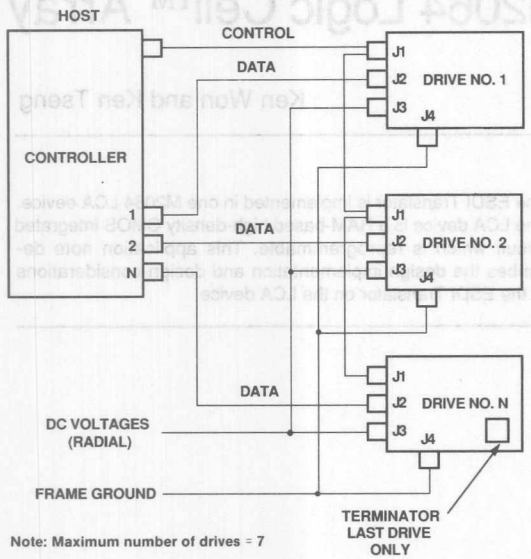


Figure 1. Connection Between the Controller and Multiple Drives

The ESDI Translator handshakes serial commands from a disk controller, deserializes the commands and passes the commands to a microcontroller. The command data word structure is shown in Figure 2.

The Command Function bits define functions to be executed by the disk drive. These functions are seek, recalibrate, request status, request configuration, select head group, control, data strobe offset, track offset, initiate diagnostics, set bytes per sector and set configuration. Some of these functions, such as the control function, have modifiers for more detailed functional description. Other commands have parameters that contain numbers. For the seek command, the parameter specifies the cylinder number that the drive will seek to. The request status and request configuration commands require data from the disk drive to be transferred back to the disk controller. In our current design, internal registers A and B in the LCA device represent the upper and lower bytes of the command respectively.

Figure 3 illustrates the relationships between the disk controller, the ESDI Translator, and the microcontroller. The PROM is used to store the configuration data for the LCA device. The Done/$\overline{\text{Program}}$ (D/$\overline{\text{P}}$) output is driven LOW when the device is being configured. Configuration data is read from the PROM device during configuration. After configuration is complete, the LCA device drives the D/P pin HIGH to deselect the PROM. Drive selection, write protection, command completion and fault detection are also handled by the ESDI Translator.

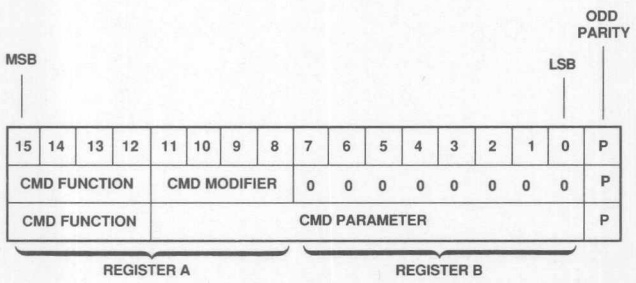


Figure 2. Command Data Word Structure

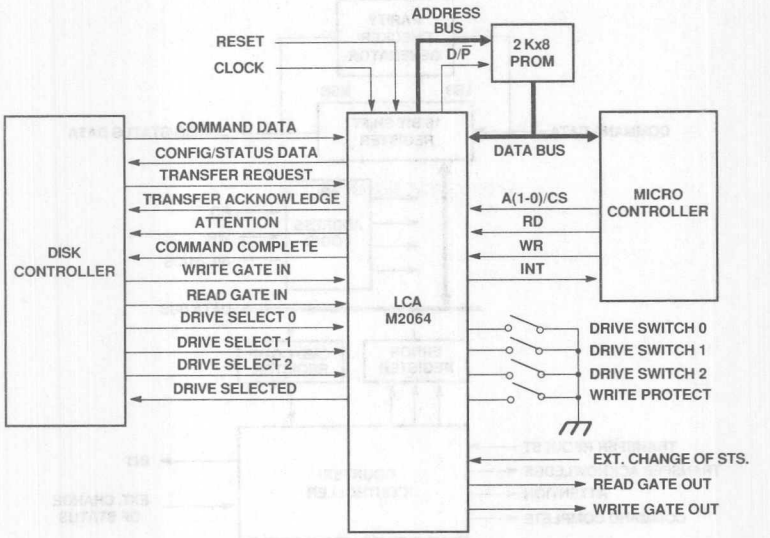


Figure 3. An ESDI Translator Implemented on the LCA Device

5

Why Use an LCA Device in an ESDI Translator

The ESDI interface standard requires more logic functions to be built into the disk drive than some other interface standards, such as the ST506 standard. However, the external dimensions of a disk drive usually have to conform to an industrial standard form factor. Thus, the use of high-density semicustom chips is the logical solution to increase functionality without increasing the external dimensions.

The LCA device is a high-density CMOS integrated circuit available from Monolithic Memories. Its high gate density allows the implementation of an ESDI Translator in a single chip. Fifteen standard SSI and MSI chips would be necessary for the same application. If PLDs are used to implement the ESDI Translator, more than one would be necessary because a large amount of logic is required, thus occupying more board space.

A major advantage in using the LCA device is that it can speed up the design cycle, enabling the manufacturer to have a shorter time-to-market. Also, many peripheral products are produced in relatively small quantities aiming at very specialized markets. The LCA device, which has no NRE cost, makes the production of small quantities more economical than the gate array. Another advantage of the LCA device over other semicustom chips, such as the gate array, is its reprogrammability feature. The LCA device is RAM-based which can easily be reprogrammed by the user in the final system. This feature is especially important in the peripheral products market, where many products have short life spans.

Design Implementation

The ESDI Translator is responsible for all control interfaces between the disk controller and the disk drive. An internal block diagram of the ESDI Translator is shown in Figure 4. It consists of five major logic building blocks:

- Drive selection
- Read gate/write gate
- Counter/controller
- Shift register and parity generator/checker
- Internal register address decoder

Drive selection on ESDI-compatible drives involves three signals from the disk controller, Drive Select 0-2. These three drive select lines are encoded so that up to seven drives may be connected to the same ESDI port, as shown in Table 1.

On the LCA device, the drive number is selected by connecting the drive switch pins to either VCC or GND. When the code on the drive select lines, DSX, equals the code on the drive switch pins, DSWX, where X may be 0, 1, or 2, the drive is selected and the Drive Selected signal, DSELD, is asserted. Once the drive has been selected, serial commands output by the disk controller will be read by the LCA device. The actual implementation is shown in Figure 5, using two CLBs and seven IOBs. BDS0 and BDS1 are the names of the CLBs in the current design. In our design, names that begin with the letter B or P are used to designate a CLB or an IOB respectively. SCLK is the system clock.

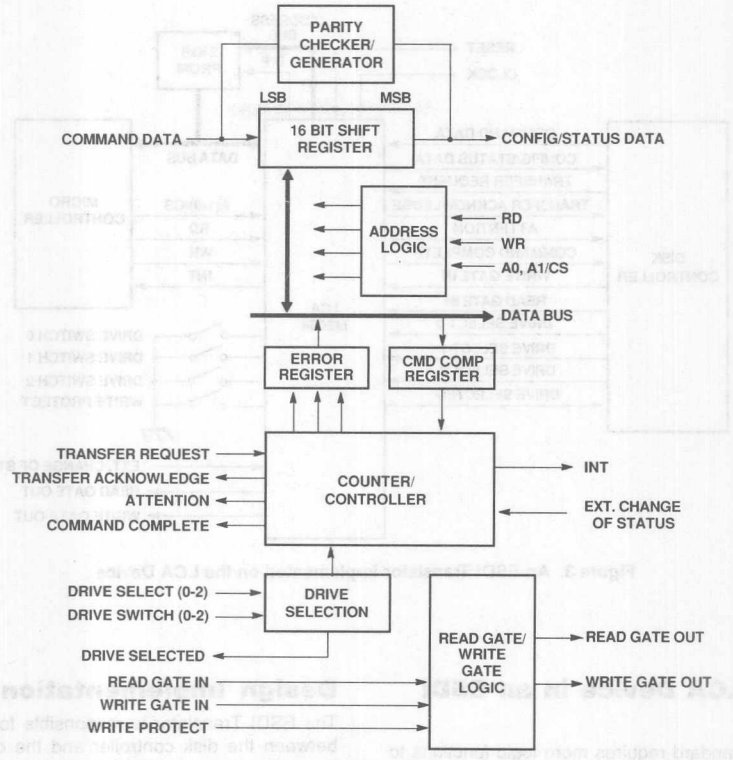


Figure 4. ESDI Translator Internal Block Diagram

DS2/ DSW2	DS1/ DSW1	DS0/ DSW0	DRIVE
0	0	0	None
0	0	1	Select Drive 1
0	1	0	Select Drive 2
0	1	1	Select Drive 3
1	0	0	Select Drive 4
1	0	1	Select Drive 5
1	1	0	Select Drive 6
1	1	1	Select Drive 7

Table 1. Drive Selection

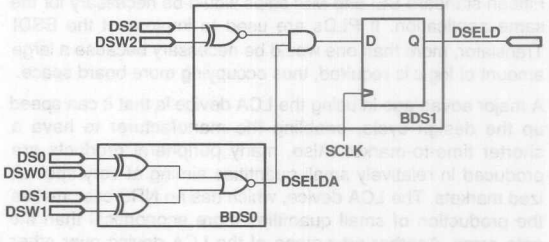


Figure 5. Configuration of the BDS0 and BDS1 CLBs to Provide the Drive Selected Signal

Figure 6 shows the configurations of the two CLBs as displayed on the computer screen by the XACT development software. In Figure 6a, the CLB is configured as one function of four variables. The D flip-flop is not used. The logic representation, truth table, Karnaugh map, signal names, block name and Boolean equation are shown.

In Figure 6b, the CLB is configured as one function of three variables, the output of which is connected to the D flip-flop. The Q output of the D flip-flop becomes the output of this CLB.

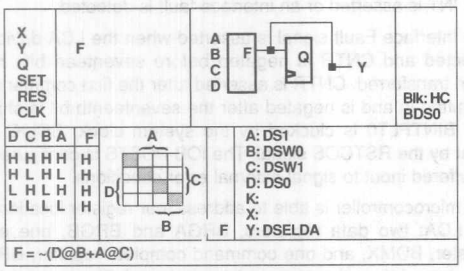


Figure 6a. Drive Selection Logic Implemented in a CLB (BDS0)

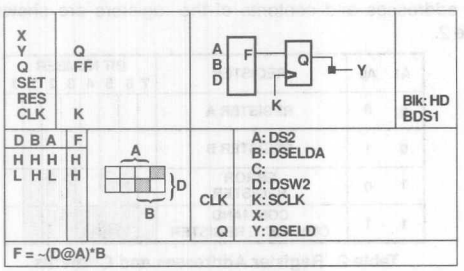


Figure 6b. Drive Selection Logic Implemented in a CLB (BDS1)

The LCA device also performs logic functions for the Read Gate and Write Gate logic blocks. The Read Gate signal allows data to be read from the disk, and the Write Gate signal allows data to be written on the disk. These signals from the disk controller are input to the LCA device as Read Gate In and Write Gate In. Under normal operating conditions, Read Gate Out is asserted when Read Gate In is asserted and Write Gate Out is asserted when Write Gate In is asserted. When both Read Gate In and Write Gate In are asserted, a write error condition results and the Attention line is asserted, signalling the disk controller that an error has occurred. Also, the LCA device may be used to provide write protection to a disk drive. When the Write Protect signal is asserted, the Write Gate Out signal will not be asserted when the Write Gate In signal is asserted, preventing the write circuitry from being activated. These logic functions are implemented in two CLBs, BRWG0 and BRWG1, as shown in Figure 7.

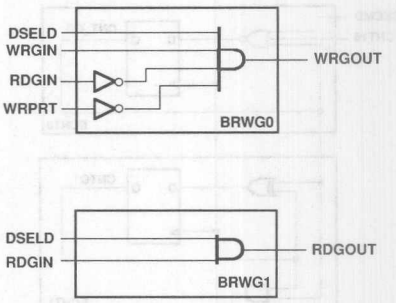


Figure 7. Read Gate/Write Gate Logic Implementation

The Counter/Controller handshakes commands from the disk controller. It also handshakes status/configuration data to the disk controller. It is also responsible for generating the Interrupt, Attention and Command Complete signals. Seventeen bit commands (one bit is parity) are transferred from the disk controller to the LCA device via the Command Data line. The serial bit transfer is performed using a pair of handshaking signals, Transfer Request (TREQ) and Transfer Acknowledge (TACK). TREQ is asserted by the disk controller when a bit is valid on the Command Data line, and TACK is asserted by the LCA device when the command bit has been read. The handshaking action is shown in Figure 8.

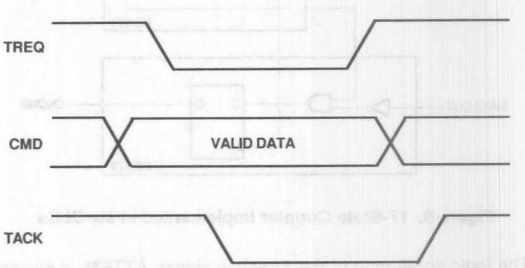


Figure 8. Command Data Transfer

A 17-state counter counts the number of command bits shifted in or shifted out of the data registers. Its implementation is shown in Figure 9. The SHIFT signal is asserted, thus incrementing the counter, whenever there is a transfer request and the LCA device is selected. CNT_Q0 is the lowest bit of the shift register counter. This bit is inverted whenever SHIFT is asserted unless sixteen bits have already been shifted into the LCA device (CNT16 asserted, or CKCMD asserted when all seventeen bits have been shifted in). The DATAOUT signal is asserted after a Request Status command or a Request Configuration command has been transferred and the internal data registers have been loaded with data to be serially shifted out. Hence, when DATAOUT is asserted, CKCMD is negated and the counter is enabled.

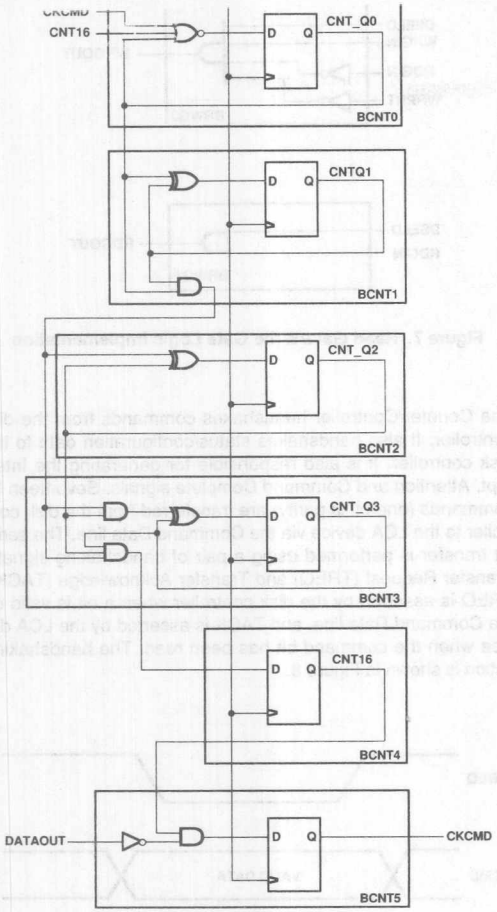


Figure 9. 17-State Counter Implemented in six CLBs

The logic generation of the Attention signal, ATTEN, is shown in Figure 10. Whenever there is a write fault, WRFLT, a parity error, PARERR, an interface fault, INTFLT, or an external error, CSTS(Change of Status), the ATTEN signal is asserted. When both WRGIN and RDGIN signals are asserted and the LCA device is selected, the WRFLT signal is asserted. The WRFLT signal is negated when the command transferred to the LCA device Register A is the Reset command defined by the ESDI standard, which has a Command Function of 0101 (Control) and a Command Modifier of 0000 (Reset Attention and Standard Status). The Command Function and Command Modifier formats are shown in Figure 2.

Three CLBs, BPG0, BPG1, and BPG2, are responsible for generating the PARERR signal. BPG0 is a multiplexer that selects the source of the input to the parity generator/checker (BPG1). If data is being shifted into the LCA device (parity checker mode), the CMDBITA input is chosen. If data is being shifted out of the LCA device (parity generator mode), the RA_Q7 output is chosen. These two signals are also shown in Figure 11. BPG1 is an odd parity generator/checker. In the parity checker mode, whenever an odd number of ones are passed through this CLB, the output is one. This output signal is connected to BPG2, which inverts the signal and asserts or negates the PARERR signal accordingly. If parity is correct, an interrupt is asserted to the microcontroller informing the microcontroller that a command has been received and is ready to be read. In the parity generator mode, the output of BPG1 is the parity bit. BPG1 is clocked by the SHIFT input, which is asserted whenever there is a transfer request and the LCA device is selected. BPG1 is reset by the Reset Parity Generator input, RSTPGEN. This signal is asserted when either INT is asserted or an interface fault is detected.

The Interface Fault signal is asserted when the LCA device is selected and CNTR is negated before seventeen bits have been transferred. CNTR is asserted after the first command bit is shifted in and is negated after the seventeenth bit is shifted out. BINTFLT0 is clocked by the system clock, SCLK, and reset by the RSTCOS signal. The IOB PCSTS is configured as a buffered input to signal external error conditions.

The microcontroller is able to address four register locations in the LCA: two data registers, BRGA and BRGB, one error register, BDMX, and one command complete register, BRC7. Only three bits in the error register are used. They are bit 0 for parity error, bit 4 for interface fault and bit 7 for write fault. Only one bit in the command complete register is used. This is bit 0, which has a value of zero when the command is completed. The addresses and contents of the registers are shown in Table 2.

A1	A0	REGISTER	BIT NUMBER 7 6 5 4 3 2 1 0
0	0	REGISTER A	
0	1	REGISTER B	
1	0	ERROR REGISTER	
1	1	COMMAND COMPLETE REGISTER	

Table 2. Register Addresses and Contents

The LCA device implementation of the registers and multiplexer is shown in Figure 11. When the interrupt signal is asserted, the microcontroller reads the two data registers by setting the A1 and A0 address lines appropriately and asserting the RD signal to the LCA device. These data registers contain the command that is transferred from the disk controller through the CMDBITA input. If the command is a request data command, configuration or status data is written to these two data registers by the microcontroller. These two bytes, plus a parity bit that is generated by the parity generator in the LCA device, are serially transferred to the disk controller over the Config/Status Data line through RA_Q7. After all seventeen bits have been transferred, the Command Complete signal is asserted. If the command is not a request data command, the microcontroller executes the command and upon completion, writes a byte of zeros to the command complete register of the LCA. When the command complete register is written with all zeros, the Command Complete signal is asserted by the LCA device.The command complete register may only be written, not read. The status register contains error bits that are set when errors are detected. This register may only be read and not written.

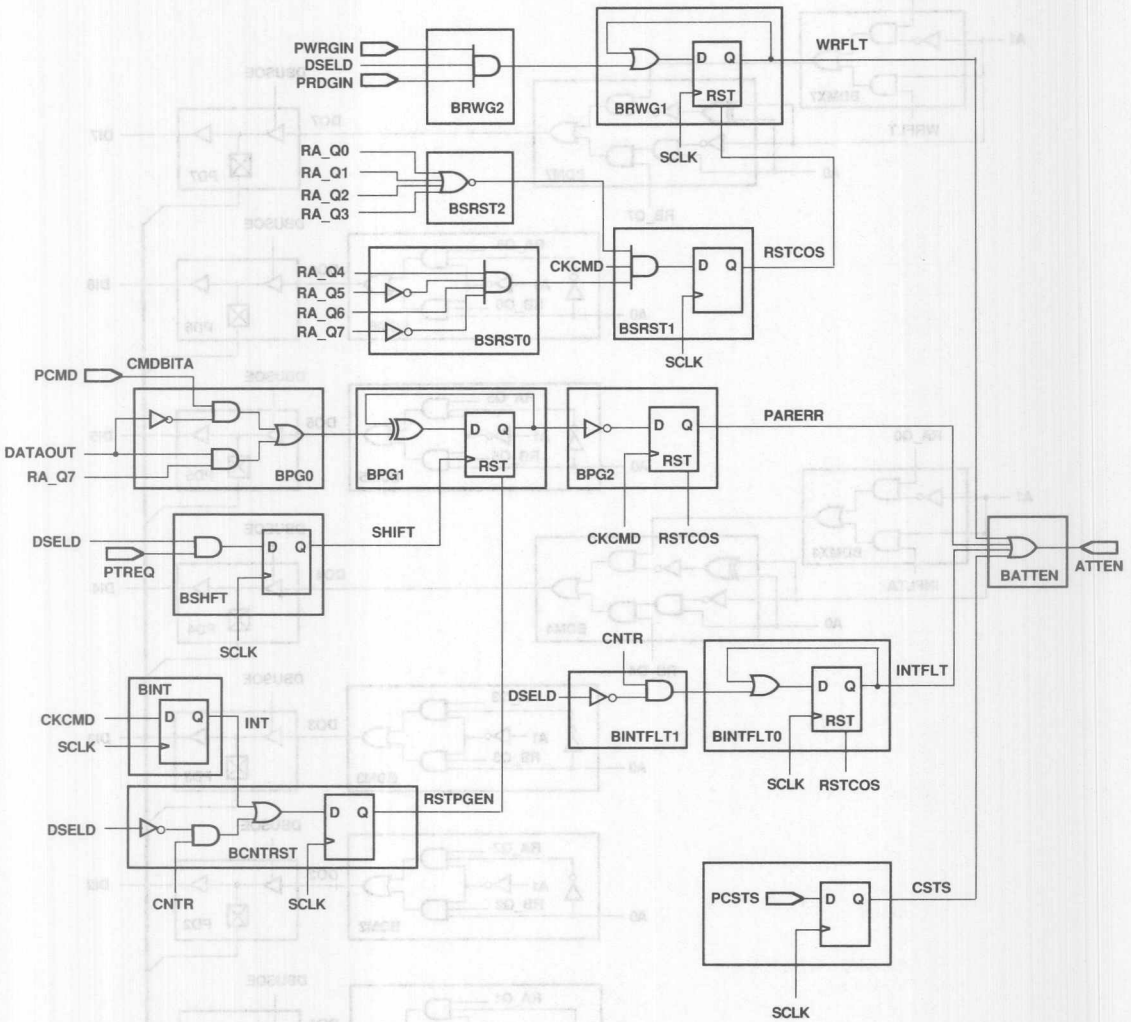


Figure 10. Generation of the Attention Signal by the Four Possible Error Conditions

5

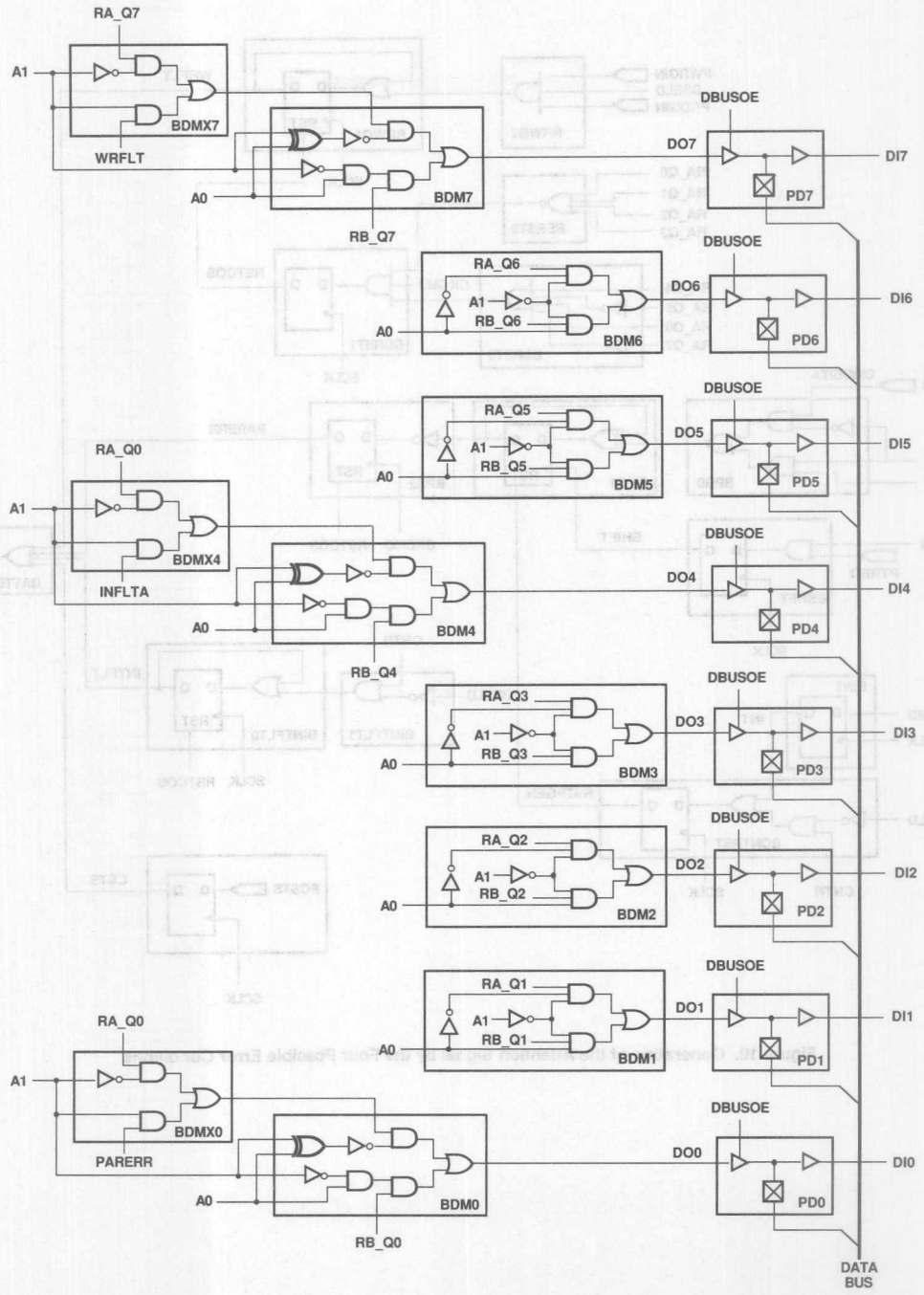


Figure 11. Registers A, B, Error Register, and Multiplexer

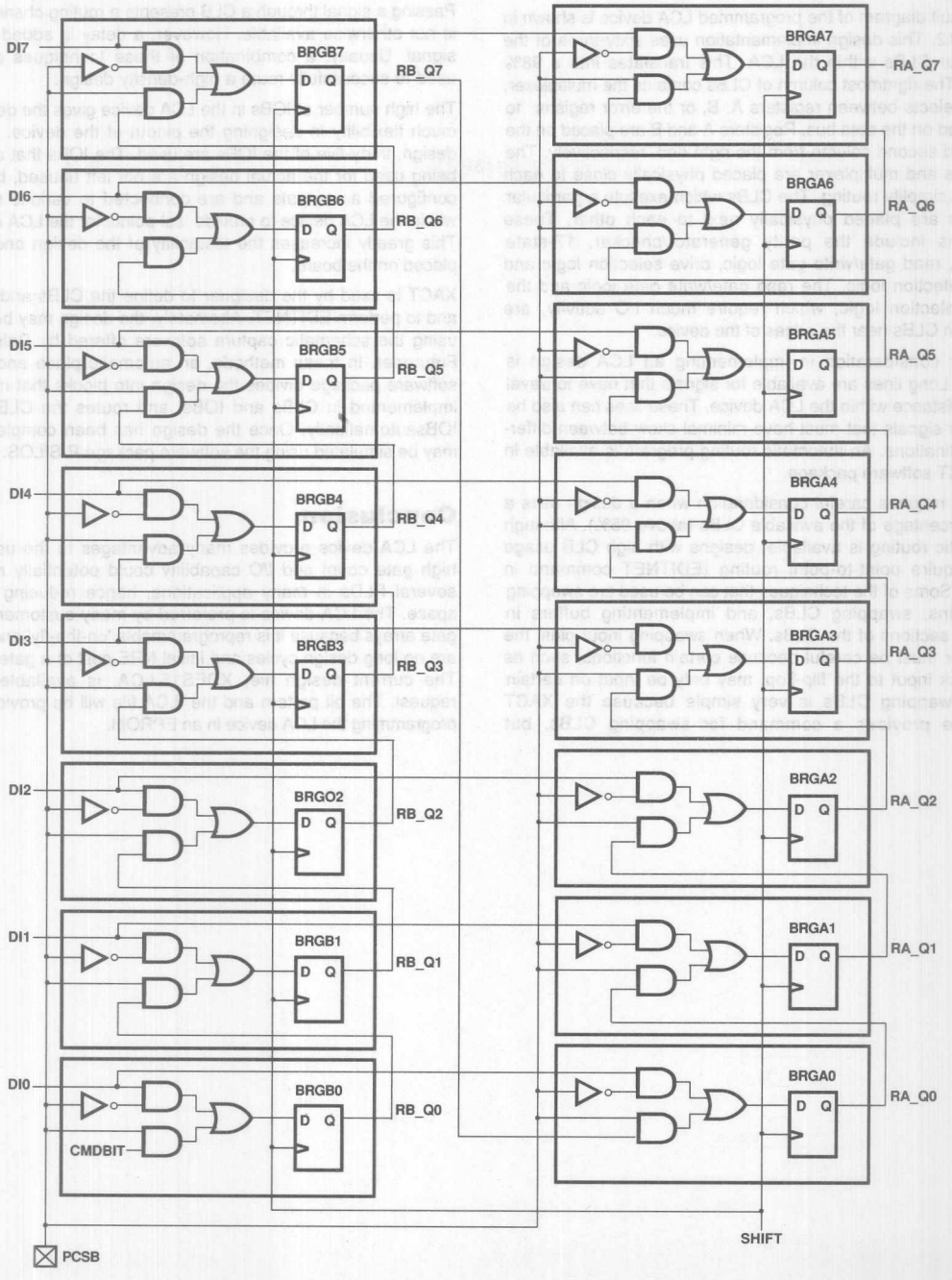


Figure 11. Registers A, B, Error Register, and Multiplexer (Continued)

Design Considerations

The circuit diagram of the programmed LCA device is shown in Figure 12. This design implementation uses sixty-three of the sixty-four CLBs within the LCA. This translates into a 98% usage. The rightmost column of CLBs contains the multiplexer, which selects between registers A, B, or the error register to be placed on the data bus. Registers A and B are placed on the third and second column from the right end, respectively. The registers and multiplexer are placed physically close to each other to simplify routing. The CLBs which execute a particular function are placed physically next to each other. These functions include the parity generator/checker, 17-state counter, read gate/write gate logic, drive selection logic and error detection logic. The read gate/write gate logic and the drive selection logic, which require much I/O activity, are placed in CLBs near the edges of the device.

Another consideration in implementing an LCA design is routing. Long lines are available for signals that have to travel a long distance within the LCA device. These lines can also be used for signals that must have minimal skew between different destinations. An automatic routing program is available in the XACT software package.

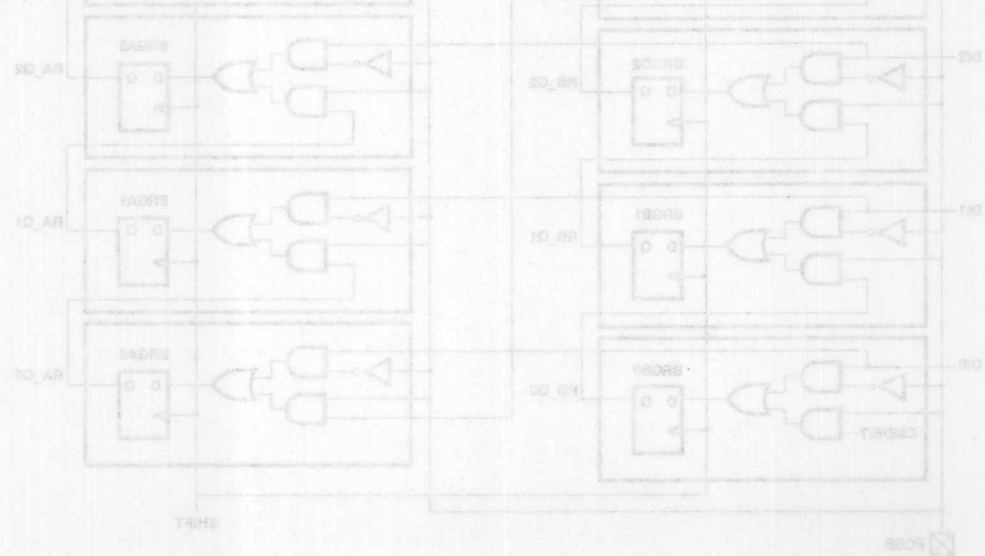
Routing requires careful consideration when a design uses a high percentage of the available CLBs (above 95%). Although automatic routing is available, designs with high CLB usage may require point-to-point routing (EDITNET command in XACT). Some of the techniques that can be used are swapping input pins, swapping CLBs, and implementing buffers in unused sections of the CLBs. When swapping input pins, the designer must be careful because certain functions, such as the clock input to the flip-flop, may only be input on certain pins. Swapping CLBs is very simple because the XACT software provides a command for swapping CLBs, but sometimes not all of the signals can be successfully rerouted. Passing a signal through a CLB presents a routing channel that is not otherwise available. However, a delay is added to the signal. Usually, a combination of these techniques can be used to successfully route a high-density design.

The high number of IOBs in the LCA device gives the designer much flexibility in designing the pinout of the device. In this design, thirty-five of the IOBs are used. The IOBs that are not being used for the actual design are not left unused, but are configured as outputs and are connected to various signals within the LCA device to provide test points for the LCA device. This greatly increases the testability of the design once it is placed on the board.

XACT is used by the designer to define the CLBs and IOBs, and to perform EDITNET. Alternately, the design may be input using the schematic capture software offered by Daisy and Futurenet. In these methods, an automatic place and route software package divides the design into blocks that may be implemented in CLBs and IOBs, and routes the CLBs and IOBsautomatically. Once the design has been completed, it may be simulated using the software package P-SILOS.

Conclusion

The LCA device provides many advantages to the user. Its high gate count and I/O capability could potentially replace several PLDs in many applications, hence reducing board space. The LCA device is preferred by many customers over gate arrays because it is reprogrammable 'on-the-fly' and there are no long design cycles and initial NRE cost of a gate array. The current design file, XDES15.LCA, is available upon request. The bit pattern and the .LCA file will be provided for programming the LCA device in an EPROM.



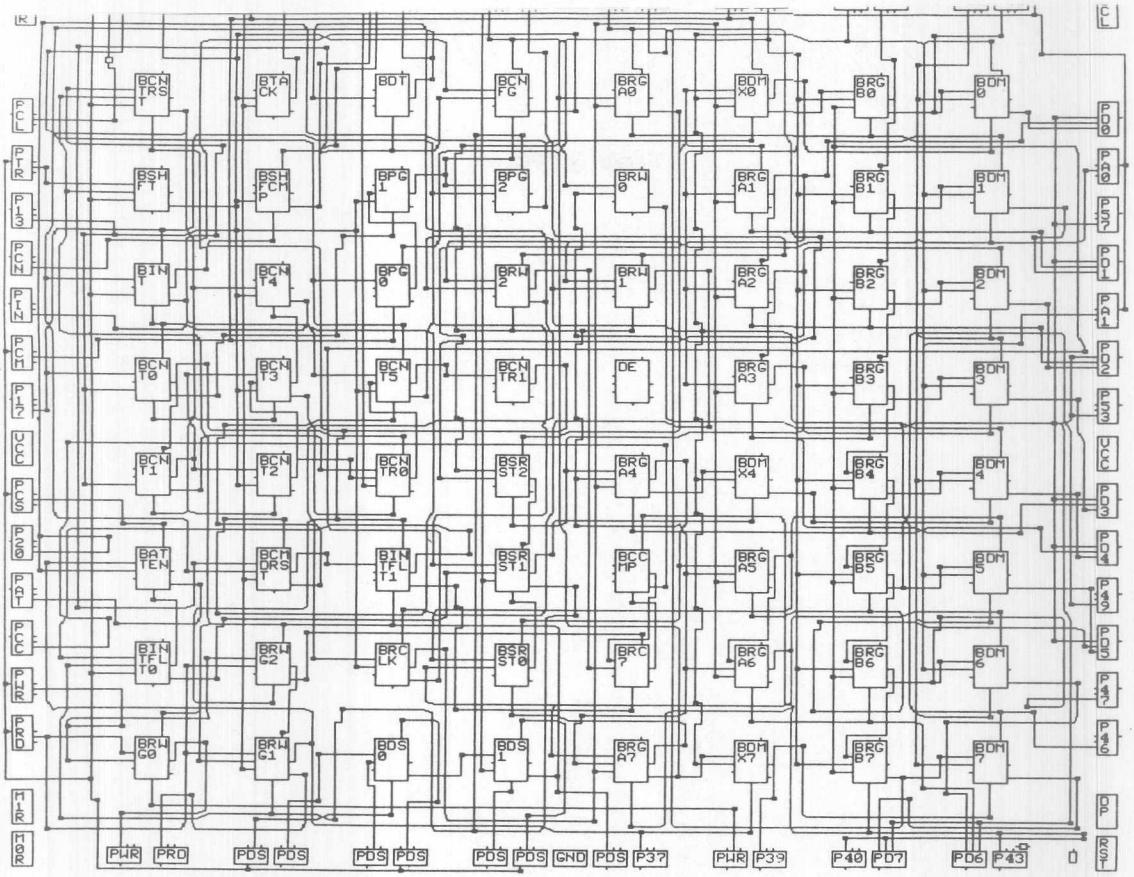


Figure 12. ESDI Translator LCA Design

5